НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення розподілених систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Аналіз поведінки автомобіля за даними з камер спостереження»

Виконав (-ла):
студент (-ка) IV курсу, групи ТВ-61
Артамонов Олексій Юрійович _____

Керівник:
доцент, к.т.н.
Шаповалова Світлана Ігорівна _____

Рецензент:
к.т.н. ст. викладач кафедри ТЕУТ та АЕС
Рачинський Артур Юрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

Артамонов Олексій Юрійович
(прізвище, ім’я, по батькові)

1. Тема роботи Аналіз поведінки автомобіля за даними з камер спостереження
керівник роботи доцент, к.т.н. Шаповалова Світлана Ігорівна
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи ”5” червня 2020 р.
3. Вихідні дані до роботи мова програмування — Python, середовище розробки — Jupyter
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) розглянути та проаналізувати набір даних про дорожньо-транспортні пригоди, візуалізувати значущі критерії набору даних, обґрунтувати вибір математичної моделі для передбачення та алгоритм розробки програмного додатку, розробити програмне забезпечення, зробити висновки за результатами роботи
5. Перелік ілюстративного матеріалу «Еволюція моніторингу дорожнього руху», «Мета», «Задачі», «Датасет», «Попередня обробка датасету», «Зменшення об’єму датасету», «Візуалізація значущих критеріїв датасету», «Розв’язання задачі прогнозування», «Засоби розробки», «Сторонні модулі», «Архітектура», «Діаграма прецедентів», «Формування скороченого датасету», «Результати візуалізації», «Результати прогнозування», «Висновки»

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”11” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	11.10.2019	
2.	Вивчення та аналіз задачі	11.10.2019-23.12.2019	
3.	Розробка архітектури та загальної структури системи	03.02.2020-04.03.2020	
4.	Розробка структур окремих підсистем	05.03.2020-12.04.2020	
5.	Програмна реалізація системи	13.04.2020-17.05.2020	
6.	Оформлення пояснювальної записки	18.05.2020-07.06.2020	
7.	Захист програмного продукту	27.05.2020	
8.	Передзахист	10.06.2020	
9.	Захист	15.06.2020	

Студент _____ _____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Мета роботи — розроблення програмного забезпечення аналізу даних щодо скоєних ДТП для прогнозування та запобігання їх у майбутньому., за допомогою використання методології Data Science та Machine Learning, та практичне оволодіння навичками та прийомами структурного аналізу даних. У процесі розробки досліджувалися методи та технології для аналізу даних, а також алгоритми прогнозування часових рядів.

Реалізовано робочу та протестовану веб-систему, яка створена для отримання результатів аналізу у вигляді візуалізованих статистичних даних значущих критеріїв вхідного датасету, створено опис процесу розробки системи та документацію до неї.

Розробка програмної системи відбувалася у середовищі Jupyter.

Обсяг роботи складає 65 сторінок, 34 ілюстрацій, 10 формул, 10 використаних джерел, 4 додатки.

Ключові слова: Data Science, Data Mining, Machine Learning, аналіз дорожньо-транспортних інцидентів, безпека дорожнього руху, ARIMA, Python, Jupyter.

ABSTRACT

The purpose of the work is to develop software for data analysis of accidents to predict and prevent them in the future, using the methodology of Data Science and Machine Learning, and practical mastery of skills and techniques of structural data analysis. In the process of development, methods and technologies for data analysis, as well as algorithms for forecasting time series were studied.

Implemented a working and tested web system, which was created to obtain the results of the analysis in the form of visualized statistics of significant criteria of the input dataset, created a description of the system development process and documentation for it.

The software system was developed in the Jupyter environment.

The volume of work is 65 pages, 34 illustrations, 10 formulas, 10 used sources, 4 appendices.

Keywords: Data Science, Data Mining, Machine Learning, traffic accident analysis, road safety, ARIMA, Python, Jupyter.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ СКОРОЧЕНЬ І ТЕРМІНІВ

Датасет — набір даних.

Data Mining — добування даних.

Data Science — наука про дані.

Machine Learning — машинне навчання.

ARIMA (Autoregressive Integrated Moving Average) — моделі авторегресії — ковзного середнього.

CART (Classification and Regression Tree) — дерева рішень у машинному навчанні.

ANN (Artificial neural networks) — штучні нейроні мережі.

ВСТУП

За даними Всесвітньої організації охорони здоров'я (ВООЗ) майже 1,25 мільйона людей гинуть у дорожніх аваріях щороку, в середньому 3 287 загиблих за день. Крім того дорожньо-транспортні катастрофи належать до 9-ої провідної причини загибелі та становлять 2,2% усіх смертей у світі. Безпека доріг є актуальною проблемою для багатьох країн, загибелі та інвалідність внаслідок дорожньо-транспортних пригод поступово визнаються основним питанням охорони здоров'я.

Поліпшення забезпечення безпеки на дорогах та зменшення кількості жертв на дорогах вимагає постійних дій на основі аналізу ґрунтовних даних. У деяких країнах зараз впроваджуються "легкі" заходи, на рівні збільшення дорожніх патрулів та камер спостереження. Однак для суттєвого зменшення кількості жертв на дорогах, включаючи серйозні травми, та запобігання ДТП необхідно заздалегідь передбачати дорожні ситуації на основі аналізу наявних даних про обставини аварій, їхню тяжкість, механізми, що призводять до аварій, дії учасників дорожнього руху тощо. Маючи достатньо обмежену інформацію про реальний дорожній рух, можна допомогти певним територіям передбачити, коли і де автомобільні аварії найімовірніше відбудуться в їх межах. Чиновники можуть використовувати цю інформацію для спрямування заходів з безпеки на ділянках дороги, де вони матимуть найбільший вплив, тому автоматизація моніторингу ситуації на дорогах для безпеки руху є сучасним трендом та має практичну значущість.

З поширенням використання обчислювальних технологій кількість даних про аварії постійно накопичується протягом останніх кількох років. Дані про нещасні випадки часто є одними з найцінніших активів, оскільки допомагають у формуванні бюджету та впровадженню нових законів, а також допомагають урядовцям приймати рішення, що стосуються планування та розвитку інфраструктури.

Однак, оскільки кількість даних щодо дорожніх обставин зростає, необхідно визначити методи, технології та інструменти пошуку для аналізу великих обсягів даних. Їх застосування допоможе зрозуміти причину зростання аварій у різних

регіонах світу. Крім цього використання такого аналітичного інструменту може допомогти у створенні бази знань щодо забезпечення безпеки на дорогах.

Наразі вже створено декілька ресурсів зі збору такої інформації. Наприклад, Міністерство Земель, Інфраструктури, Транспорту та Туризму Японії надає “Інформацію про транспортну статистику” в Інтернеті. За допомогою цієї статистики дослідники аналізують тенденції та прогнозують майбутній стан транспортних мереж.

В межах ЄС існує ряд науково-дослідних проектів, таких як SafetyNet, Dacota та SafetyCube, які збирають інформацію про дорожній рух з різних джерел, аналізують її та допомагають владі регулювати бюджети на дорожньо-транспортну галузь, а також вводити інші міри для забезпечення безпеки на дорогах. Також аналізом дорожньо-транспортних пригод займається Інститут метрики та оцінки здоров’я (IHME). IHME — науково-дослідний інститут, що працює в галузі глобальної статистики охорони здоров’я та оцінки впливу. Він збирає дані, пов’язані зі здоров’ям, з усіх доступних джерел. IHME розробляє інноваційні аналітичні інструменти для відстеження тенденцій смертності, захворювань та факторів ризику, а також відображає багато результатів своїх досліджень у візуалізації даних. Всі ці приклади демонструють, що методи і засоби Data Mining використовуються на практиці та являються актуальними.

1. ЗАДАЧА АНАЛІЗУ ПОВЕДІНКИ АВТОМОБІЛЯ ЗА ДАНИМИ З КАМЕР СПОСТЕРЕЖЕННЯ

1.1. Постановка задачі

Дана робота спрямована на аналіз набору даних(далі датасет) про дорожні інциденти. На виході користувач отримує набір інтуїтивно зрозумілих даних, щодо кількості інцидентів за кожний рік або місяць, їх локації, в якому штаті було найбільше та найменше інцидентів у графічному вигляді та ін., а також прогнозовані місця наступних аварій з можливою кількістю дорожньо-транспортних пригод на декілька місяців вперед.

Для розроблення відповідного програмного забезпечення потрібно розв’язати наступні задачі:

1. Виконати попередню обробку датасету: трансформувати категоріальні ознаки, виконати очистку набору даних.
2. Виокремити та візуалізувати значущі критерії датасету.
3. Провести аналіз набору математичних моделей для здійснення найточнішого прогнозування.
4. Створити механізм прогнозування, провести його навчання математичну модель на основі обробленого набору даних та надати отримані результати у зрозумілому вигляді.

Для аналізу в даній роботі обрано датасет “Us Accidents” щодо безпеки дорожнього руху, які були зібрані в 49 штатах США у період з лютого 2016 року по грудень 2019. Цей датасет являє собою файл формату *.csv, який містить набір характеристик інцидентів. Дані про аварії збиралися за допомогою декількох постачальників даних. Так два API, які надають потокові дані про ДТП та трафік, отримані різними структурами, такими як державні департаменти транспорту,

правоохоронні органи, камери дорожнього руху та датчики руху в межах дорожніх мереж. Наразі в цьому наборі даних присутні близько 3,0 мільйонів записів аварій. Кожен запис про нещасний випадок складається з різноманітних внутрішніх та контекстуальних атрибутів, таких як місцеположення, час, опис на природній мові, погода, період дня та інші корисні характеристики.

Цей датасет є надто громіздким та має деяку кількість характеристик, непотрібних для поточного аналізу. Ці характеристики суттєво впливають на продуктивність та швидкість обробки даних. Тому один з етапів вирішення поставленої задачі є видалення характеристик, які мають велику кількість пустих значень або колонок які є копією іншої колонки лише з відмінністю у одиницях вимірювання. Надалі саме цей, зменшений датасет буде вхідною інформацією для аналізу.

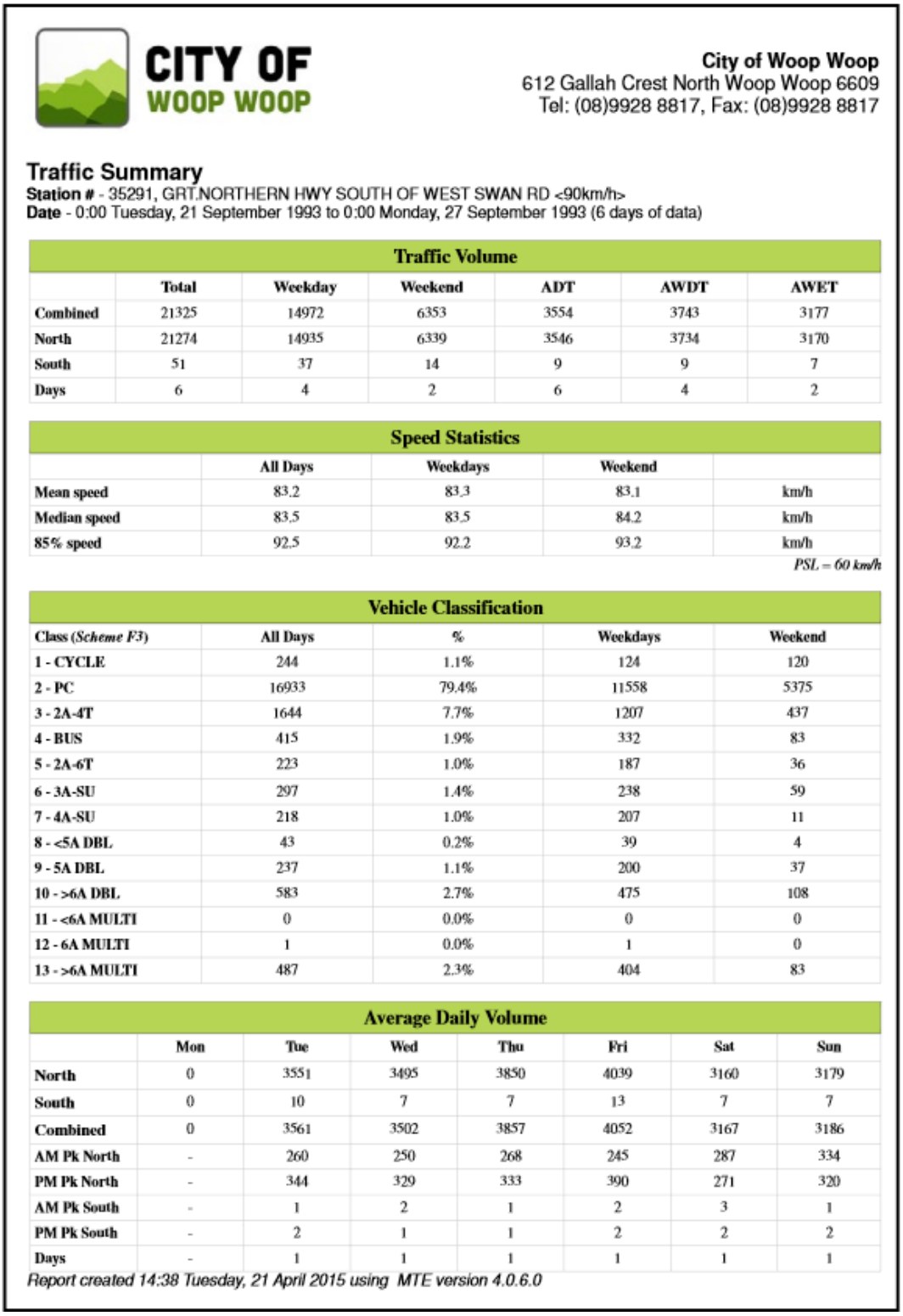
Метою роботи є розроблення програмного забезпечення для демонстрації результатів аналізу та механізмів запобігання ДТП за допомогою використання методології Data Science та Machine Learning.

1.2. Існуючі рішення

Задача моніторингу дорожнього руху є актуальною, а тому на ринку програмного забезпечення існує багато програмних продуктів для її вирішення. Наприклад, системи під назвами MetroCount Traffic Executive та HERE Traffic Analytics.

MetroCount Traffic Executive — це програмне забезпечення для аналізу трафіку. Головна функція цієї системи, окрім аналізу, це формування звітів. Поряд зі стандартними звітами, спеціальні звіти можуть бути створені відповідно до організації. На рисунку 1.1 демонструється звіт, який містить інформацію про обсяги, швидкості та класифікацію транспортних засобів. Класифікація використовується для

моніторингу деградації доріг, прогнозування тривалості життя дороги та визначення тенденцій на дорогах. MTE має алгоритми для класифікації транспортних засобів, а також включає розробку персональних схем, розроблених на замовлення клієнта. Стандартний звіт зображений на рисунку 1.2, демонструє загальний обсяг транспортних засобів у погодинно розбитий на кожен тиждень.



CITY OF WOOP WOOP

City of Woop Woop
612 Gallah Crest North Woop Woop 6609
Tel: (08)9928 8817, Fax: (08)9928 8817

Traffic Summary
Station # - 35291, GRT.NORTHERN HWY SOUTH OF WEST SWAN RD <90km/h>
Date - 0:00 Tuesday, 21 September 1993 to 0:00 Monday, 27 September 1993 (6 days of data)

Traffic Volume

	Total	Weekday	Weekend	ADT	AWDT	AWET
Combined	21325	14972	6353	3554	3743	3177
North	21274	14935	6339	3546	3734	3170
South	51	37	14	9	9	7
Days	6	4	2	6	4	2

Speed Statistics

	All Days	Weekdays	Weekend	
Mean speed	83.2	83.3	83.1	km/h
Median speed	83.5	83.5	84.2	km/h
85% speed	92.5	92.2	93.2	km/h

PSL = 60 km/h

Vehicle Classification

Class (*Scheme F3*)	All Days	%	Weekdays	Weekend
1 - CYCLE	244	1.1%	124	120
2 - PC	16933	79.4%	11558	5375
3 - 2A-4T	1644	7.7%	1207	437
4 - BUS	415	1.9%	332	83
5 - 2A-6T	223	1.0%	187	36
6 - 3A-SU	297	1.4%	238	59
7 - 4A-SU	218	1.0%	207	11
8 - <5A DBL	43	0.2%	39	4
9 - 5A DBL	237	1.1%	200	37
10 - >6A DBL	583	2.7%	475	108
11 - <6A MULTI	0	0.0%	0	0
12 - 6A MULTI	1	0.0%	1	0
13 - >6A MULTI	487	2.3%	404	83

Average Daily Volume

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
North	0	3551	3495	3850	4039	3160	3179
South	0	10	7	7	13	7	7
Combined	0	3561	3502	3857	4052	3167	3186
AM Pk North	-	260	250	268	245	287	334
PM Pk North	-	344	329	333	390	271	320
AM Pk South	-	1	2	1	2	3	1
PM Pk South	-	2	1	1	2	2	2
Days	-	1	1	1	1	1	1

Report created 14:38 Tuesday, 21 April 2015 using MTE version 4.0.6.0

Рисунок 1.1 — Приклад звіту по трафіку у програмі MTE

Site: 35291.0.0N
Description: GRT.NORTHERN HWY SOUTH OF WEST SWAN RD <90km/h>
Filter time: 13:00 Monday, 20 September 1993 => 14:24 Monday, 27 September 1993
Scheme: Vehicle classification (VRX)
Filter: Cls(1-12, 14-15) Dir(N) Sp(10,160) Headway(>0) Span(0 - 100)

	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Averages	
	20 Sep	21 Sep	22 Sep	23 Sep	24 Sep	25 Sep	26 Sep	1 - 5	1 - 7
Hour									
0000-0100	*	10	14	10	19	18	33	13.3	17.3
0100-0200	*	6	6	4	11	20	17	6.8	10.7
0200-0300	*	10	6	12	7	11	9	8.8	9.2
0300-0400	*	9	7	7	8	15	9	7.8	9.2
0400-0500	*	22	18	16	13	14	12	17.3	15.8
0500-0600	*	53	61	63	66	51	30	60.8	54.0
0600-0700	*	130	141	159	148	117	56	144.5	125.2
0700-0800	*	246	222	248	215	149	98	232.8	196.3
0800-0900	*	260	250	262	226	194	120	249.5	218.7
0900-1000	*	210	227	230	246	230	211	228.3	225.7
1000-1100	*	235	211	222	244	278	334	228.0	254.0
1100-1200	*	205	244	269	226	289	274	236.0	251.2
1200-1300	*	222	187	243	213	274	320	216.3	243.2
1300-1400	239	219	197	244	246	260	286	229.0	241.6
1400-1500	255	213	219	256	240	203	218	236.6	229.1
1500-1600	332	346	331	336	349	212	280	338.8	312.3
1600-1700	309	311	319	319	350	232	254	321.6	299.1
1700-1800	297	309	315	295	392	183	206	321.6	285.3
1800-1900	179	217	214	217	281	132	153	221.6	199.0
1900-2000	97	120	104	164	244	93	92	145.8	130.6
2000-2100	67	88	80	104	107	63	67	89.2	82.3
2100-2200	62	60	68	104	72	42	49	73.2	65.3
2200-2300	43	55	53	59	99	47	46	61.8	57.4
2300-2400	30	25	21	34	46	53	17	31.2	32.3
Totals									
0700-1900	*	**2993**	**2936**	**3141**	**3228**	**2636**	**2754**	**3059.9**	**2955.4**
0600-2200	*	**3391**	**3329**	**3672**	**3799**	**2951**	**3018**	**3512.6**	**3358.7**
0600-0000	*	**3471**	**3403**	**3765**	**3944**	**3051**	**3081**	**3605.6**	**3448.5**
0000-0000	*	**3581**	**3515**	**3877**	**4068**	**3180**	**3191**	**3720.1**	**3564.6**
AM Peak	*	**0800**	**0800**	**1100**	**0900**	**1100**	**1000**		
	*	**260**	**250**	**269**	**246**	**289**	**334**		
PM Peak	*	**1500**	**1500**	**1500**	**1700**	**1200**	**1200**		
	*	**346**	**331**	**336**	**392**	**274**	**320**		

* - No data.

Рисунок 1.2 — Приклад звіту по щотижневим підрахункам кількості транспортних засобів у програмі MTE

Весь датасет також може бути коротко представлений за один усереднений тиждень. Представлення класифікації в контексті швидкостей забезпечує метод виявлення специфічних проблем щодо швидкості руху. Завдяки загальному формату файлів у всіх лічильниках MetroCount, програмне забезпечення MTE дозволяє підводити підсумки та фільтрувати різні наборів даних, включаючи дані про велосипедистів та пішоходів. Ця функція може бути використана для підсумовування потоків трафіку через лінійні графіки або порівняння обсягів режиму руху. MTE

включає функції управління обстежень, щоб забезпечити належне покриття мережі регулярно контрольованими сайтами.

MTE включає функціональність для автоматизації аналізу декількох наборів даних. Можна скласти стандартизовані звіти в одному сценарії обстеження трафіку та застосувати їх до майбутніх обстежень трафіку для послідовного аналізу трафіку. MTE розроблений для роботи на комп'ютерах з операційною системою Windows. Остання версія MTE також може аналізувати набори даних, записані в 1990-х. Інформація про рух транспортних засобів представляє корисну інформацію для моніторингу працездатності, особливо стосовно перевантаженості та максимальної продуктивності. Програмне забезпечення MTE забезпечує декілька варіантів виводу, включаючи XLS, CSV та інші формати бази даних, приклад виводу зображений на рисунку 1.3. Інструменти налаштування дозволяють користувачам додавати та видаляти необхідну статистику із звітів.

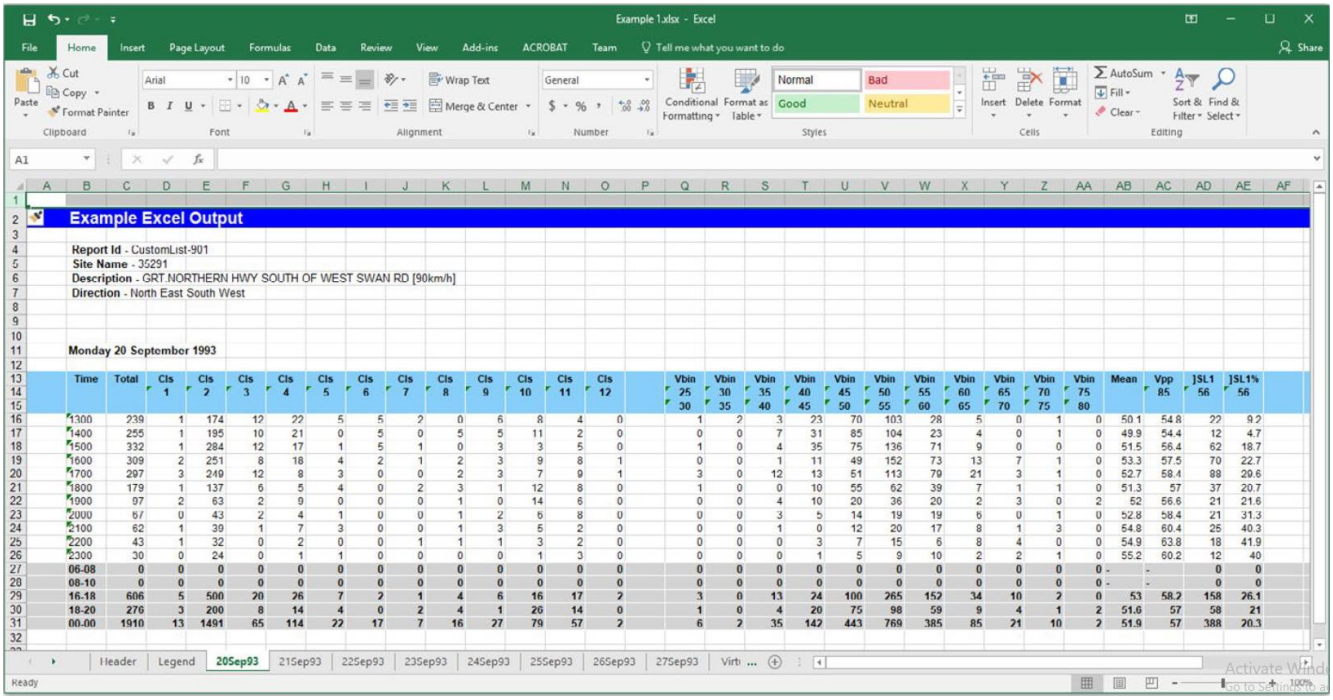


Рисунок 1.3 — Приклад виводу звіту у форматі XLS

HERE Traffic Analytics — інтерактивний онлайн-інструмент, який візуалізує в режимі реального часу очікувані рівні заторів протягом дня у понад 180 містах світу. Крім того, на його інформаційній панелі, що зображена на рисунку 1.4, представлена мобільність міста, яка демонструє, як далеко люди можуть очікувати поїзду від центру міста протягом 5, 10, 15 або 20 хвилин, залежно від умов руху в конкретний

час та може надавати водіям можливість виявити оптимальний час для початку своїх подорожей. У той же час транспортні органи влади можуть бачити, як впливає рух транспорту на місто.

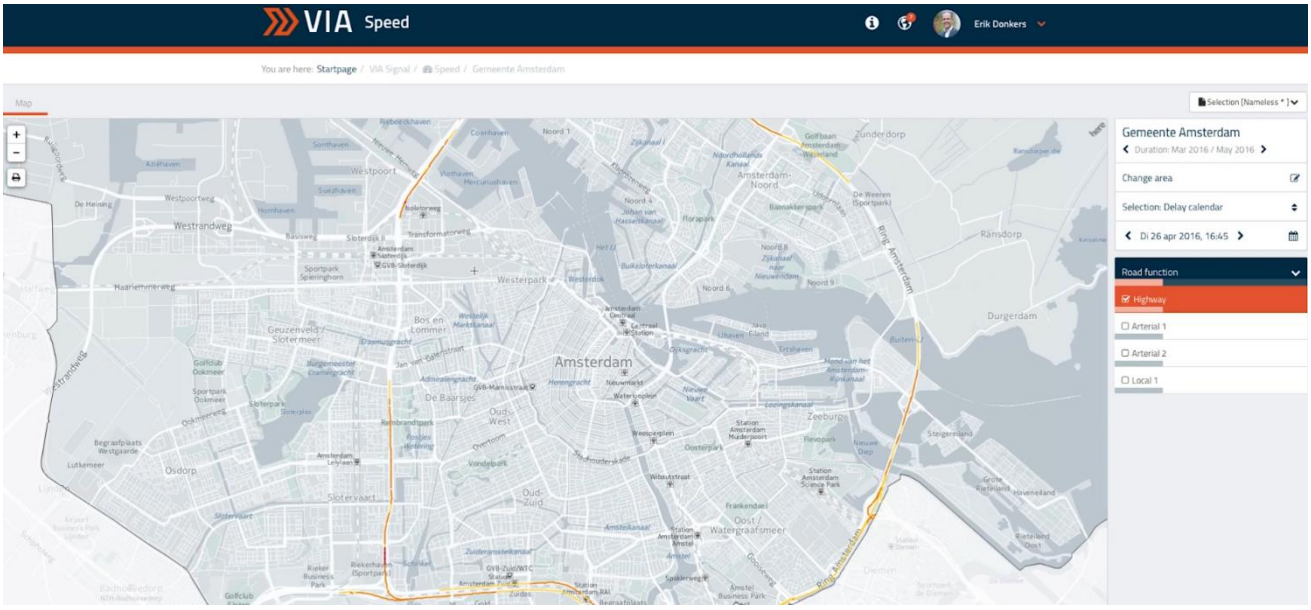


Рисунок 1.4 — Інформаційна панель веб-застосунку HERE Traffic Analytics

Також, цей програмний продукт дозволяє клієнтам зрозуміти, що відбувається на проїжджих дорогах, завдяки потужній інтелектуальній інформації про місцезнаходження та переглянути, наприклад, середню швидкість по місту у вибраний клієнтом період часу, як це зображено на рисунку 1.5. Це дозволяє продукувати більш обізнане прийняття рішень щодо дорожнього будівництва, потоку руху, управління дорожньою мережею та землекористування.

Ця система, використовує дані з великої кількості зондів GPS, завдяки цьому агенції та підприємства можуть зрозуміти наслідки змін, перш ніж вони відбудуться, та керувати трафіком з більшою точністю. Для своїх служб, HERE збирає інформацію з великої кількості GPS-датчиків щодня та використовує понад 100 різних джерел інцидентів. Також, ця система використовує дані датчиків транспортних засобів які збираються у реальному часі від декількох марок автомобілів.

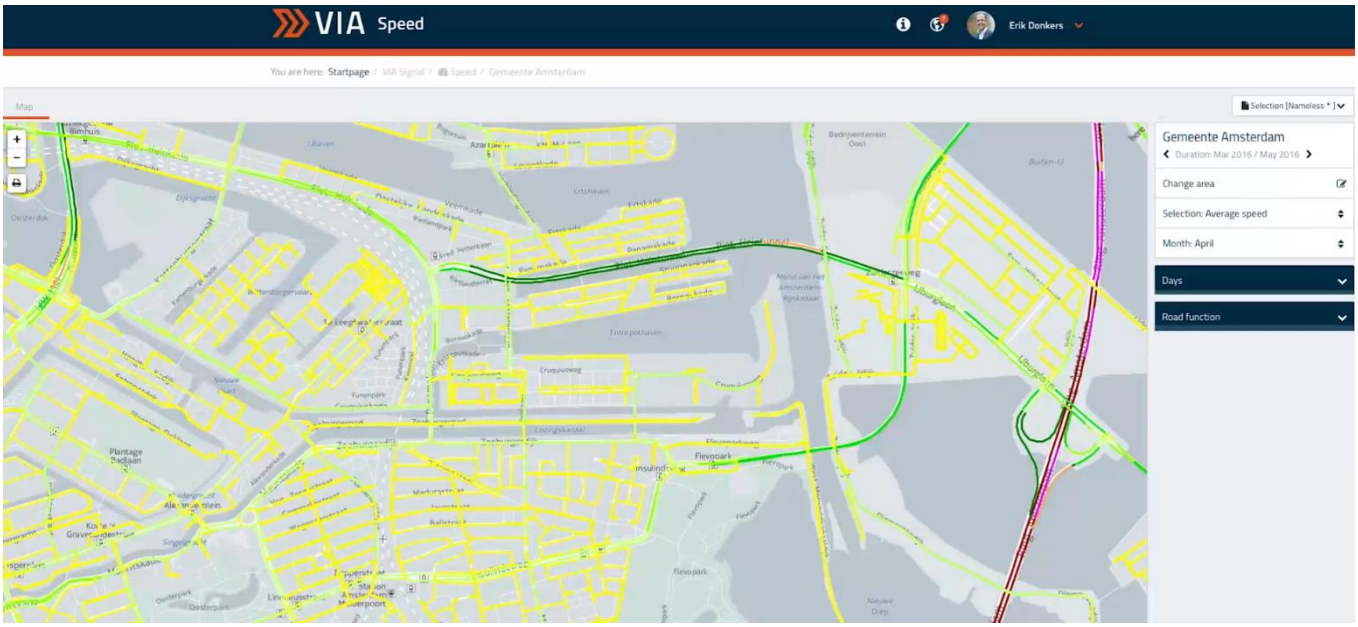


Рисунок 1.5 — Демонстрація середньої швидкості за квітень на інформаційній панелі

Після збору, HERE обробляє дані через платформу HERE Open Location — постійно оновлюючи дані кожні 60 секунд. Завдяки спеціалізованим сервісам, побудованим на цій інформації, HERE може допомогти водіям, органам дорожнього руху та бізнесу бути більш поінформованими про час подорожі та дорожні умови.

Також, ця система надає API для розробників, яке включає в себе інформацію про трафік, яка збирається у багатьох країнах світу, доступ до даних трафіку в реальному часі з можливістю їх отримання в форматах XML або JSON, включаючи інформацію про швидкість і перевантаженість для регіону, визначеного в кожному запиті. API також може надавати додаткові дані, такі як геометрія ділянок дороги щодо потоку.

Усі розглянуті рішення задачі моніторингу дорожнього руху мають один великий недолік, який полягає в тому, що це комерційні системи, доступ до яких відкриється тільки після придбання. Також, жодний з цих програмних продуктів не виконує аналіз дорожньо-транспортних пригод та детальної візуалізації статистичних даних про них. Всі ці мінуси говорять про те, що ринок систем моніторингу дорожнього руху не повністю покриває запити користувачів та потребує розширення.

2. ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.1. Мова програмування Python

Data Science, як сфера, завоювала велику популярність за останні кілька років. Інструменти, які використовуються для аналізу даних, а саме: програмування SAS, програмування R, Hadoop, Python, SQL та інші.

Python — це один із таких інструментів, що має унікальну характеристику, він є мовою програмування загального призначення, простою у користуванні, що стосується аналітичних та кількісних обчислень. Python був у галузі досить тривалий час і застосовується в таких сферах, як наукові обчислення, нафта, газ, фінанси, фізика, обробка сигналів тощо. Python також використовувався в побудові додатків, таких як “YouTube”, був важливим для внутрішньої інфраструктури “Google” тощо. Цей інструмент може координувати масивні кластери серверів комп'ютерної графіки та допомагати у виробництві фільмів. Цей привабливий атрибут — це те, чому від цього залежать такі компанії, як “Disney”, “Sony”, “Dreamworks” тощо.

Що стосується Data Science, Python — це дуже потужний інструмент, який є відкритим і гнучким, що додає йому більшої популярності. Data Science передбачає екстраполяцію корисної інформації з масивних сховищ статистики, реєстрів та даних. Ці дані, як правило, несортовані і їх важко співвідносити з будь-якою значущою точністю. Machine Learning може створювати зв'язки між різними наборами даних, але вимагає серйозної обчислювальної потужності.

Python заповнює цю потребу, будучи мовою програмування загального призначення. Це дозволяє створити вихід CSV для зручного зчитування даних в електронній таблиці. Крім того, більш складні вихідні файли, які можуть бути використані кластерами Machine Learning для обчислення.

Machine Learning може допомогти скласти більш точні прогнозні моделі на основі минулих подій. Python може це зробити, оскільки він легкий і ефективний при

виконанні коду, але він також багатофункціональний. Також Python може підтримувати об'єктно-орієнтовані, структуровані та функціональні стилі програмування.

Зараз в Python налічується понад 70 000 бібліотек, і ця кількість продовжує зростати. Відомо, що є маса бібліотек для маніпулювання даними, і їх надзвичайно просто вивчити та використовувати для всіх аналітиків даних. Крім незалежної платформи, Python має можливість легко інтегруватися в наявну інфраструктуру, а також може розв’язувати найскладніші проблеми. Цей інструмент потужний, простий і добре взаємодіє з іншими, крім того, працює всюди.

Як вже було сказано раніше, Python пропонує багато бібліотек, орієнтованих на Data Science. Незалежно від того, що вчені прагнуть зробити з Python, будь то прогнозована причинно-наслідкова аналітика чи будь-яка інша, Python має набір інструментів для виконання різноманітних потужних функцій. Це не дивно, чому вчені його використовують.

Python все ще знаходиться на стадії розробки, тобто він отримує регулярні оновлення та випуски. По мірі того, як Big Data та Machine Learning стають все більш поширеними в бізнесі та уряді.

2.2. Середовище розробки Jupyter

Jupyter — це середовище розробки інтерактивних документів (блокнотів). Блокноти дозволяють викладачам та студентам поєднувати разом обчислювальну інформацію (код, дані, статистику) з розповідними, мультимедійними та графічними. Факультет університету може використовувати його для створення інтерактивних підручників, наповнених поясненнями та прикладами, які студенти можуть перевірити прямо у своїх браузерах. Студенти можуть використовувати це, щоб пояснити свої міркування, показати свою роботу та встановити зв’язки між своєю класною роботою та світом за її межами. Вчені, журналісти та дослідники можуть

використовувати їх для відкриття своїх даних, обміну історіями, що стоять за їхніми обчисленнями, та подальшої співпраці та інновацій.

Jupyter був створений для того, щоб легше було показати свою програмістську роботу і дозволити іншим приєднатися до неї [8]. Jupyter дозволяє об’єднувати код, коментарі, мультимедіа та візуалізації в інтерактивному документі — блокноті, який можна спільно використовувати, повторно використовувати і повторно працювати.

А оскільки Jupyter працює через веб-браузер, сам блокнот може бути розміщений на вашому локальному комп’ютері або на віддаленому сервері. Спочатку розроблений для додатків науки про дані (Data Science), написаних на Python, R і Julia, Jupyter корисний у всіх видах проектів. Більшість людей вперше знайомляться із Jupyter за допомогою візуалізації даних. Він може містити візуалізацію деякого набору даних у вигляді графічного зображення. Jupyter дозволяє додавати авторські візуалізації, а також спільно використовувати їх і дозволяти інтерактивні зміни в коді і наборі даних.

Хмарні сервіси, такі як GitHub та Pastebin, надають способи спільного використання коду, але вони в основному не інтерактивні. За допомогою Jupyter можна переглядати код, виконувати його та відображати результати безпосередньо у веб-браузері як зображено на рисунку 2.1.

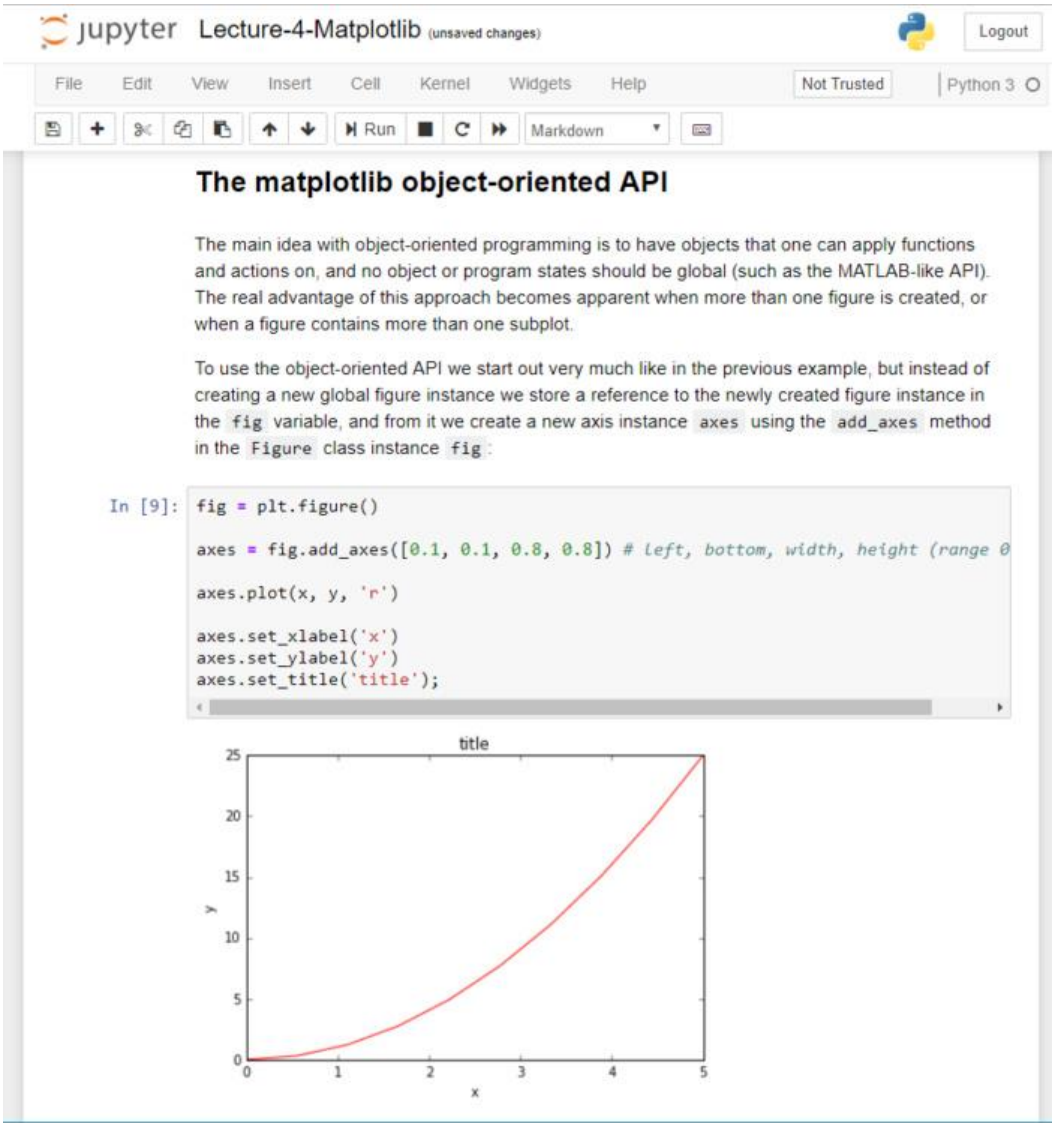


Рисунок 2.1 — Інтерфейс Jupyter

Код блокноту Jupyter (Jupyter Notebook) — не є статичним, його можна редагувати та повторно запускати поступово в режимі реального часу, зворотний зв’язок надається безпосередньо в браузері. В блокноти також можна вбудовувати інтерактивні елементи керування (наприклад, повзунки або поля для введення тексту), які можна використовувати як джерела вхідних даних для коду.

Якщо є фрагмент коду, який треба пояснити рядок за рядком, як він працює, зі зворотним зв’язком у реальному часі, можна вбудувати його в Jupyter Notebook. Найкраще, що код залишатиметься повністю функціональним — його можна буде додати разом із поясненням, одночасно демонструвати та розповідати.

Блокноти Jupyter можуть включати в себе кілька видів елементів, кожен з яких організовано в окремі блоки

1. Текст та HTML. Звичайний текст або текст, помічений як Markdown для створення HTML, можна вставити в документ у будь-якій точці. CSS-стиль також може бути включений вбудований або доданий до шаблону, використовуваного для створення блокноту.

2. Код і його вихід. Код у блокнотах Jupyter, як правило, є кодом Python, хоча можна додати підтримку інших мов у середовищі Jupyter, таких як R або Julia. Результати виконаного коду з'являються відразу після блоків коду, і блоки коду можуть бути виконані, повторно виконані в будь-якому порядку та скільки завгодно разів.

3. Візуалізації. Графіки та діаграми можуть бути створені з коду за допомогою модулів, таких як Matplotlib, Plotly або Bokeh. Як і вихід, ці візуалізації відображаються в рядку поруч із кодом, який їх генерує. Однак код також може бути налаштований так, щоб записувати їх у зовнішні файли, якщо це необхідно.

4. Мультимедіа. Оскільки Jupyter побудований на веб—технології, він може відображати всі типи мультимедіа, що підтримуються на веб—сторінці. Можна включити їх у блокнот як HTML-елементи або ж їх можна програмно генерувати за допомогою модуля IPython.display.

5. Дані. Дані можуть бути надані в окремому файлі поряд із файлом .ipynb, що є блокнотом Jupyter, або їх можна імпортувати програмно, наприклад, включивши у блокнот код для завантаження даних із загальнодоступного Інтернет-сховища або для доступу до нього через підключення до бази даних.

Найбільш поширені випадки використання для Jupyter Notebook — це Data Science, математика та інші дослідницькі проекти, які передбачають візуалізацію даних або формул. Однак, окрім таких, є й багато інших випадків використання. Люди часто діляться результатами візуалізації даних як статичним зображенням, але це корисно лише до певного моменту. Ділячись блокнотом Jupyter, цільова аудиторія може зануритися і взаємодяти з ним. Вони можуть отримати глибоке розуміння даних, інтерактивно. Також, багато програмістів, які ведуть блог про свій досвід програмування, записують свої пости за допомогою Jupyter Notebook. Інші можуть завантажити свій блокнот і відтворити роботу. Більшість документації для модулів

Python є статичними; Jupyter Notebook можна використовувати як інтерактивну платформу для вивчення того, як працює модуль. Будь-який модуль Python, який добре працює в інтерфейсі блокнота — хороший кандидат для цього.

2.3. Бібліотека Pandas

Pandas — це NoSQL база даних в пам’яті, яка має SQL-подібні конструкції, підтримку статистики та аналітики даних, а також можливість побудови графіків [5]. Оскільки Pandas створений поверх Cython, він має менші витрати на пам’ять та працює швидше. Багато людей використовують Pandas для заміни Excel, виконання ETL, обробки табличних даних, завантаження файлів CSV або JSON тощо. Попри те, що він виріс із фінансового сектору (для аналізу даних часових рядів), тепер це бібліотека маніпулювання даними загального призначення.

Один з ключів до розуміння Pandas — розуміння моделі даних [1]. В основі Pandas лежать три структури даних, такі як: Series, DataFrame та Panel, зображені на рисунку 2.2.

Найбільш широко використовувані структури даних — це Series та DataFrame, які відповідно обробляють дані масиву та таблиці. Аналогія зі світом електронних таблиць ілюструє основні відмінності між цими типами. DataFrame схожий на аркуш із рядками та стовпцями, тоді як Series схожа на один стовпець даних. Panel — це група аркушів. Аналогічно, у Pandas, Panel може мати багато структур типу DataFrame, кожен з яких у свою чергу може мати декілька серій.

І Series, і DataFrame мають спільні функції, такі як, наприклад, індекс. Крім того, оскільки DataFrame можна розглядати як сукупність стовпців, які насправді є об’єктами Series, важливо спочатку провести всебічний огляд Series.

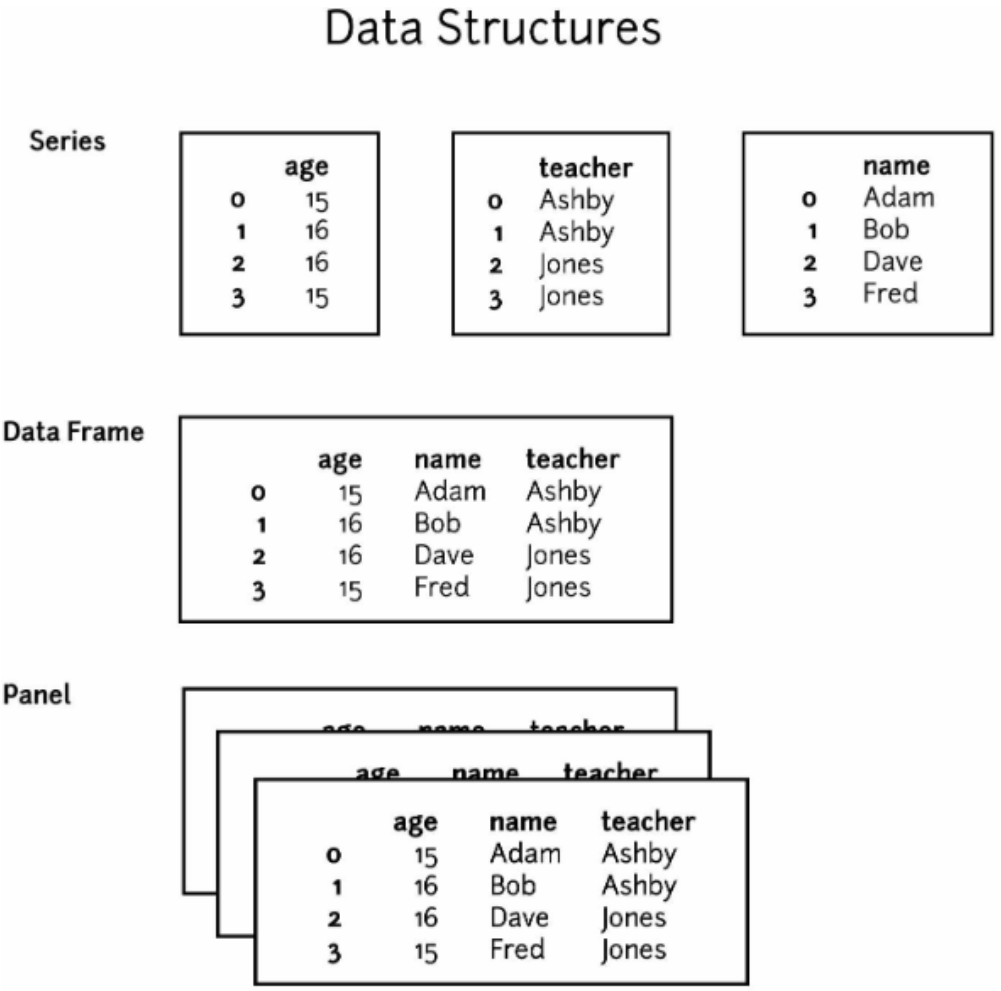


Рисунок 2.2 — Візуалізація структур даних у Pandas

Series використовується для моделювання одновимірних даних, схожих до списку у Python. Об’єкт Series також має ще кілька біт даних, включаючи індекс та ім’я. Загальна ідея Pandas — це поняття осі. Оскільки Series є одновимірною, вона має одну вісь — індекс. На рисунку 2.3 зображений приклад Series.

Artist	Data
0	145
1	142
2	38
3	13

Рисунок 2.3 — Таблиця з кількості пісень у виконавців

Щоб представити ці дані в чистому Python, можна використовувати структуру даних, аналогічну словнику зі списком точок даних, що зберігаються під ключем “дані”. Окрім запису у словнику фактичних даних, є явний запис відповідних значень

індексу даних (під ключем “індекс”), а також запис для імені даних (під ключем “ім’я”) що зображено на рисунку 2.4.

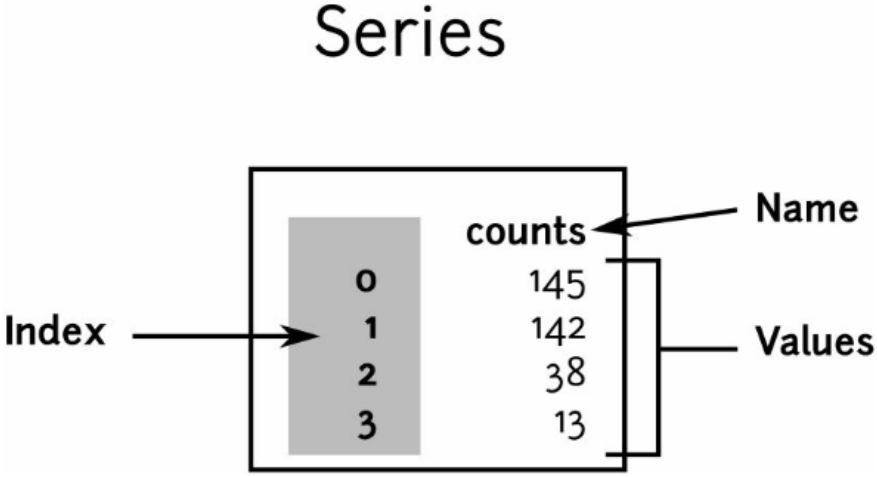


Рисунок 2.4 — Візуальне представлення Series

Крайній лівий стовпчик — це стовпчик індексу, який містить записи для індексу. Загальна назва для індексу — вісь, а значення індексу — 0, 1, 2, 3 — називаються мітками осі. Двовимірна структура в Pandas — DataFrame — має дві осі, одну для рядків та іншу для стовпців.

Крайній правий стовпчик на рисунку містить значення рядків. У цьому випадку вони є цілими числами, але в загальних значеннях Series можуть містити дані типів string, float, bool або object (довільні об’єкти Python). Щоб отримати найкращу швидкість (таку як у векторизованих операцій), значення повинні бути одного типу, хоча це не обов’язково. Значення індексу за замовчуванням — це монотонно зростаючі цілі числа.

Об’єкт Series має багато атрибутів та методів які використовуються для аналізу даних. Загалом, методи повертають новий об’єкт Series. Більшість методів повернення нового екземпляра також мають параметр inplace або copy. Це тому, що поведінка за замовчуванням має тенденцію до незмінності, а ці необов’язкові параметри за замовчуванням False та True, відповідно.

Загальні арифметичні операції для Series перевантажені для роботи як зі скалярами, так і з іншими об’єктами Series. Додавання зі скаляром (припускаючи числові значення в Series) просто додає скалярне значення до значень об’єкту Series.

Об’єкт Series також має основні методи для роботи зі статистикою, наприклад метод .sum, який обчислює суму значень у Series. Також, якщо треба обчислити середнє квадратичне значення (“очікуване значення” або середнє значення) і медіану (“середнє” значення в 50%, що відокремлює нижчі значення від верхніх значень) треба використати методи .mean або .median об’єкту Series. Обидва ці методи ігнорують NaN (якщо для параметру skipna встановлено значення False). Для не нормальних розподілів медіана корисна в якості сумарних показників. Вона більш стійка до викидів. Крім того, кількісні заходи можуть бути використані для прогнозування значення 50% (за замовчуванням) або будь-якого бажаного рівня, наприклад 10-го та 90-го перцентилів. Кількісний розрахунок за замовчуванням повинен бути дуже схожим на медіану. Щоб отримати гарне загальне враження від Series, метод .describe представляє велику кількість підсумкової статистики та повертає результат у вигляді об’єкту Series. На виході він видає кількість значень, їх середнє значення, стандартне відхилення, мінімальне і максимальне значення, а також 25%, 50% і 75% квантилі.

Двовимірний аналог одновимірних Series — це DataFrame. Якщо вважати що DataFrame орієнтований на рядки, інтерфейс покаже зворотне. Багато табличних структур даних орієнтовані на рядки. Можливо, це пов’язано з електронними таблицями та файлами CSV, які обробляються по рядкам. DataFrame, часто використовується в аналітичних цілях, і його краще розуміти, якщо розглядати його як орієнтований на стовпці, де кожен стовпчик — це Series, як зображено на рисунку 2.5.

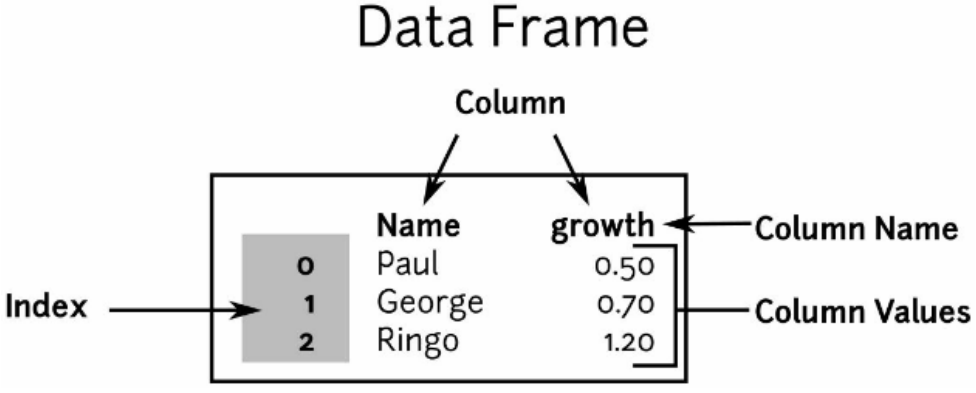


Рисунок 2.5 — Візуальне представлення DataFrame

Сказане вище повинно містити підказки, чому Series висвітлювалася настільки детально. Коли задіяні операції стовпців, часто використовується метод серії. Крім того, поведінка індексу в обох структурах даних однакова.

DataFrame можна створити з багатьох типів вхідних даних:

— стовпці (словники списків);

— рядки (список словників);

— CSV-файл (pd.read_csv);

— від NumPy.ndarray;

— І багато іншого: SQL, HDF5 тощо.

На відміну від Series, які мають одну вісь, DataFrame має дві осі, що продемонстровано на рисунку 2.6. Їх зазвичай називають віссю 0 і 1, або віссю рядка / вказівника та віссю стовпців, відповідно.

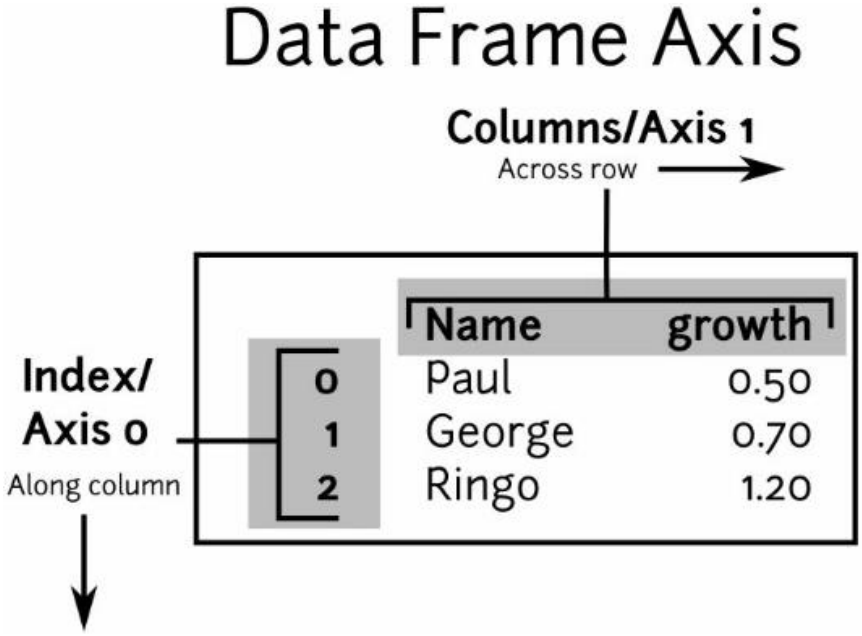


Рисунок 2.6 — Демонстрація напрямку осей у об’єкті DataFrame

Для отримання основної інформації про об’єкт використовується метод .info. Метод .info підсумовує типи та стовпці DataFrame. Він також дає зрозуміти, скільки споживається пам’яті. Коли використовуються більші набори даних, ця інформація корисна для того, щоб побачити, як використовується пам’ять. Перетворення типів рядків у числові чи типові дати може значно покращити використання пам’яті. DataFrame підтримує трансляцію арифметичних операцій. Якщо додати число до DataFrame, кожну клітинку можна збільшити на цю суму. Але є застереження:

збільшити кожне числове значення на десять, просто додавши десять до DataFrame — не вдасться. DataFrame можуть серіалізуватися у багато різноманітних форм. Найважливіша функціональність — це, мабуть, перетворення у файл CSV та з нього, оскільки цей формат є мовою даних “lingua franca”. Функція pd.read_csv створює DataFrame з CSV-файлу, а для запису у CSV, треба використати метод .to_csv.

Проектуючи надійні, прості у використанні структури даних, які знаходяться в одному ряду з іншою частиною наукового стека Python, можна зробити Python переконливим вибором для додатків аналізу даних. Pandas забезпечує міцний фундамент, на якому можна створити дуже потужну екосистему для аналізу даних.

2.4. Бібліотека NumPy

NumPy — це модуль для Python. Назва є абревіатурою для “Numeric Python” або “Numerical Python”. Це модуль розширення для Python, здебільшого написаний на C. Це гарантує, що попередньо складені математичні, числові функції та функції Numpy гарантують високу швидкість виконання.

Крім того, NumPy збагачує мову програмування Python потужними структурами даних, реалізуючи багатовимірні масиви та матриці. Ці структури даних гарантують ефективні обчислення з матрицями та масивами. Реалізація спрямована навіть на величезні матриці та масиви, які відомі під заголовком “Big Data”. Крім того, модуль постачає велику бібліотеку математичних функцій високого рівня для роботи над цими матрицями та масивами.

Масиви зберігаються в пам'яті як суміжні блоки, однакового розміру та типу. Це надає не тільки швидкий доступ, але дозволяє використовувати масиви різного розміру разом. В Python широко використовуються списки, контейнери загального призначення, які прості у використанні, але можуть містити об'єкти різного типу. Цикли Python повільніше, ніж у C.

Існує багато бібліотек, які використовують NumPy, хоча декілька, як правило, в комплекті з ним: SciPy, MatPlotLib, pandas, sympy і nose. Зокрема, NumPy та SciPy - це дві сторони монети. NumPy базується на двох модулях Python, що займаються масивами. Один з них — числовий. Numeric — це як NumPy модуль Python для високопродуктивних чисельних обчислень, але він є застарілим на сьогоднішній день. Ще один попередник NumPy — це Numarray, який є повною копією Numeric, але також є застарілим. NumPy — це злиття цих двох модулів, тобто він заснований на коді Numeric та особливостях Numarray.

NumPy містить досить багато лінійних функцій алгебри, хоча вони повинні бути в SciPy. Також SciPy пропонує більш повнофункціональні версії модулів лінійної алгебри, а також багато інших числових алгоритмів. Для наукових обчислень з Python, найкраще встановити і NumPy, і SciPy.

Ще одна відмінність полягає в тому, що NumPy написаний на C, тоді як більша частина SciPy — це тонкий шар коду, що знаходиться над науковими процедурами, які є у вільному доступі в Netlib у C та Fortran.

Безумовно, що стосується лінійної алгебри, ndarray в NumPy дозволяє робити скалярний добуток двох матриць, а також матричний добуток і підвищувати матрицю до потужності. Він може вирішити тензорні рівняння та три різні типи інверсії матриці.

NumPy має функції для фінансових обчислень, індексації, лінійної алгебри, математичних функцій, поліномів, випадкової вибірки, статистики, бінарних операцій, логічних, сортування, пошуку та операцій з рядками.

Універсальні функції, також відомі як ufuncs, — це функції, застосовані до кожного елемента вхідного масиву, при цьому результат зберігається у відповідному вихідному масиві однакового розміру.

2.5. Бібліотека Seaborn

Seaborn — це бібліотека візуалізації даних у Python на базі matplotlib. Щоб почати роботу з нею, знадобиться встановити її в терміналі за допомогою pip, або за допомогою conda. А далі просто імпортувати Seaborn як sns на початку файлу python.

Seaborn прагне зробити візуалізацію центральною частиною вивчення та розуміння даних. Функції побудови графіків оперують фреймами даних та масивами, що містять цілий dataset і внутрішньо виконують необхідну агрегацію та статистичну адаптацію моделей для створення інформативних графіків. Цілі Seaborn схожі на цілі R-ggplot, але для нього потрібен інший підхід з імперативним та об'єктно-орієнтованим стилем, який робить його простим для побудови складних сценаріїв. Якщо matplotlib “намагається зробити легкі речі легкими та важкими речами можливими”, Seaborn прагне також зробити чітко визначений набір важких речей. Часто при візуалізації статистичних даних все, що потрібно, — це побудувати гістограми та спільні розподіли змінних, що зображені на рисунку 2.7. Якщо потрібно змінити або фон, або кольори всіх графіків, це можна легко зробити за допомогою двох команд: sns.set_style та sns.set_palette.

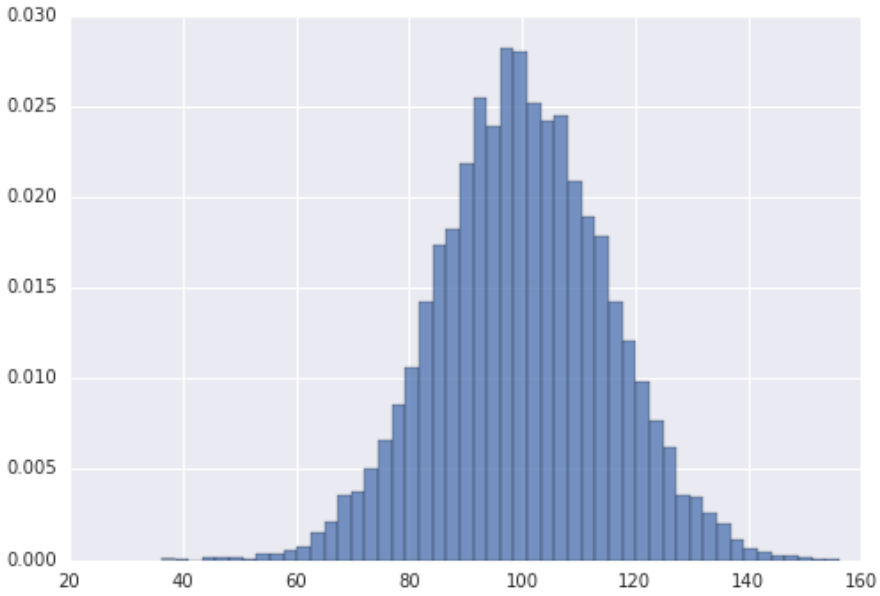


Рисунок 2.7 — Вигляд гістограми у Seaborn

Інша велика перевага Seaborn полягає в тому, що у нього є кілька вбудованих видів графіків, яких у matplotlib немає. Більшість із них можна врешті реплікувати, завдяки matplotlib, але вони не вбудовані та потребують значно більше коду і часу. Граничні та регресійні — лише два приклади графіків, для створення яких за допомогою matplotlib потрібно набагато більше часу; графік регресії демонструє регресійну лінію, довірчий інтервал та графік розсіювання, все це результат виклику однієї функції: sns.regplot. Наприклад, діаграма з довірчими інтервалами продемонстрована на рисунку 2.8.

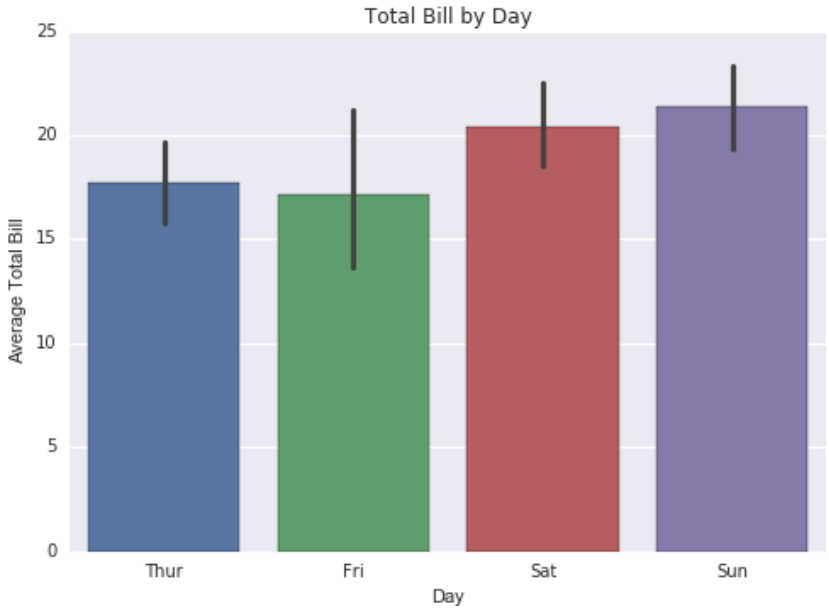


Рисунок 2.8 — Демонстрація діаграм з довірчими інтервалами

Також, Seaborn має такі види графіків як Countplot, Boxplot, Violinplot та багато інших.

Countplot — графік, який в основному рахує категорії та повертає їхню кількість. Це один із найпростіших графіків, що надається цією бібліотекою, він зображений на рисунку 2.9. Дивлячись на цей рисунок, можна сказати, що кількість чоловіків більше, ніж кількість жінок у наборі даних. Оскільки він повертає лише підрахунок на основі категоріального стовпця, потрібно вказати лише параметр x.

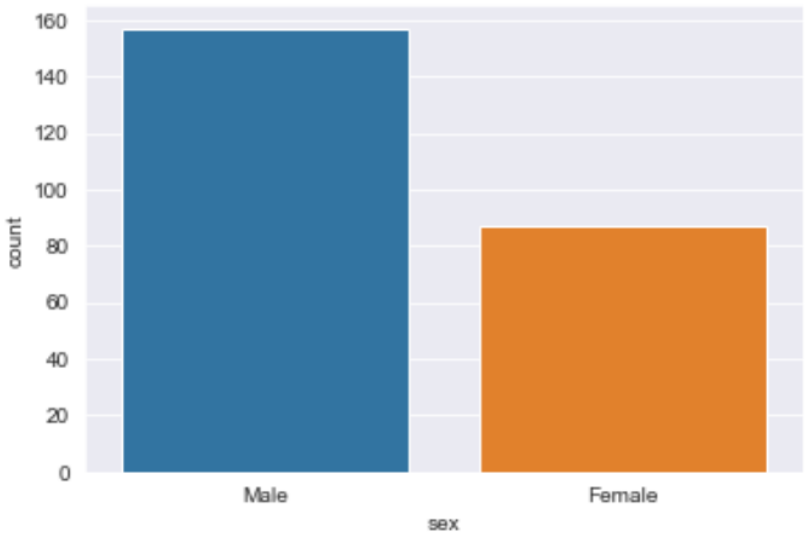


Рисунок 2.9 — Демонстрація графіку типу Countplot

Boxplot іноді називають коробковим графіком. Він показує розподіл кількісних даних, що представляють собою порівняння між змінними. Boxplot показує квартилі набору даних, тоді як вусики розширюються, щоб показати решту розподілу, тобто точки, що вказують на наявність залишків. Функція для побудови такого виду графіку приймає у аргумент x — категоричний стовпчик, а у аругмент y - числовий стовпчик. Отже, можна бачити загальний витрачений рахунок щодня. Параметр “hue” використовується для подальшого додавання категоріального поділу.

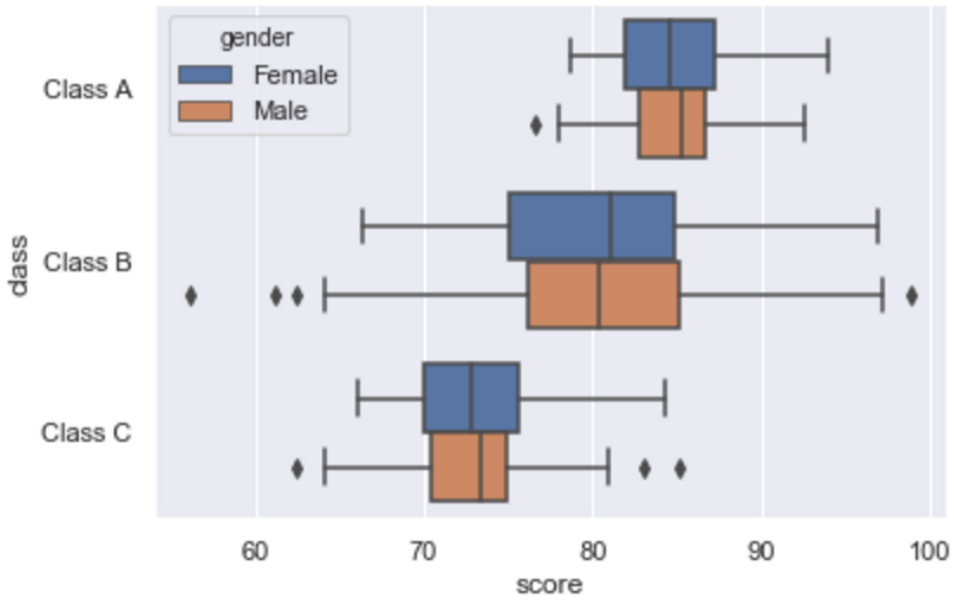


Рисунок 2.10 — Демонстрація графіку типу Boxplot

Violinplot схожий на boxplot за винятком того, що він забезпечує більш високу, більш досконалу візуалізацію і використовує оцінку щільності ядра, щоб дати кращий опис про розподіл даних, він зображений на рисунку 2.11.

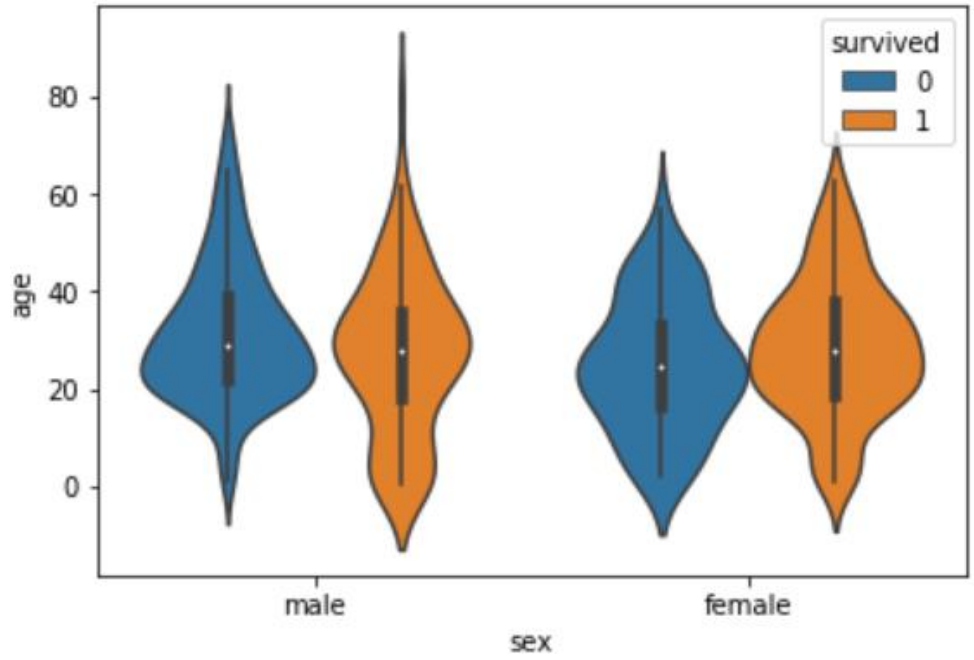


Рисунок 2.11 — Демонстрація графіку типу Violinplot

Отже, Seaborn дуже потужний інструмент візуалізації, який користується великим попитом завдяки своїй легкості та широкому переліку можливих видів візуалізації даних і тому був використаний у даній роботі, як основний.

2.6. Бібліотека Scikit-Learn

Існує кілька бібліотек Python, які забезпечують ґрунтовну реалізацію цілого ряду алгоритмів машинного навчання. Одним з найвідоміших є Scikit-Learn — пакет, що забезпечує ефективні версії великої кількості загальних алгоритмів. Scikit-Learn характеризується чистим, рівномірним та спрощеним API, а також дуже корисною та повною онлайн-документацією. Перевага цієї рівномірності полягає в тому, що, зрозумівши основне використання та синтаксис Scikit-Learn для одного типу моделі, перехід на нову модель або алгоритм стає дуже простим.

API Scikit-Learn розроблений з урахуванням наступних керівних принципів, викладених у документі API Scikit-Learn:

1. Послідовність: Усі об'єкти мають спільний інтерфейс, створений з обмеженого набору методів, з послідовною документацією.

2. Перевірка: Усі задані значення параметрів виставляються в якості загальнодоступних атрибутів.

3. Обмежена ієрархія об'єктів: Тільки алгоритми представлені класами Python; набори даних представлені в стандартних форматах (NumPy масиви, Pandas DataFrames, SciPy матриці), а назви параметрів використовують стандартні рядки Python.

4. Склад: Багато завдань машинного навчання можуть бути виражені послідовностями більш фундаментальних алгоритмів, і Scikit-Learn використовує це, де це можливо.

5. Чутливі значення за замовчуванням: Коли моделі вимагають визначених користувачем параметрів, бібліотека визначає відповідне значення за замовчуванням.

Кожен алгоритм машинного навчання в Scikit-Learn реалізований за допомогою Estimator API, який забезпечує послідовний інтерфейс для широкого кола застосунків машинного навчання.

У Scikit-Learn кожен клас моделі представлений класом Python. Наприклад, якщо треба обчислити просту модель лінійної регресії, можна просто імпортувати клас лінійної регресії.

Залежно від класу моделі, з яким ведеться робота, може знадобитися відповісти на одне або кілька питань, таких як:

— Чи треба відповідати зміщенню (тобто, у-перехоплення)?

— Потрібно, щоб модель нормалізувалася?

— Чи треба попередньо обробити функції, щоб додати гнучкість моделі?

— Яку ступінь регуляризації потрібно використати у моделі?

— Скільки компонентів моделі потрібно використати?

Це приклади важливих варіантів, які необхідно зробити після вибору класу моделі. Ці варіанти часто представляються як гіперпараметри або параметри, які

повинні бути встановлені до того, як модель підходить до даних. У Scikit-Learn гіперпараметри вибираються шляхом передачі значень при інстанціюванні моделі. Ми дослідимо, як можна кількісно мотивувати вибір гіперпараметрів у гіперпараметрах та валідації моделі.

Тепер настав час. Застосувати нашу модель до даних можна методом .fit() моделі. Ця функція викликає деяку кількість внутрішніх обчислень, залежних від моделі, і результати цих обчислень зберігаються в атрибутах, характерних для моделі, які користувач може досліджувати.

Одне питання, яке часто виникає, стосується невизначеності таких внутрішніх параметрів моделі. Загалом Scikit-Learn не пропонує інструментів для висновків із самих внутрішніх параметрів моделі: інтерпретація параметрів моделі набагато більше питання статистичного моделювання, ніж питання машинного навчання. Машинне навчання швидше фокусується на тому, що передбачає модель.

Після того як модель навчена, головне завдання контрольованого машинного навчання — це оцінити її на основі того, що вона говорить про нові дані, які не входили до навчального набору. У Scikit-Learn це можна зробити за допомогою методу .predict().

Функціональні можливості, які пропонує Scikit-Learn, включає в себе: регресію (включаючи лінійну та логістичну регресію), класифікацію (включаючи K-найближчих сусідів), кластеризація (включаючи K-Means та K-Means++), попередню обробку (включаючи нормалізацію Min-Max).

3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

В розділі наведено алгоритмічне та математичне забезпечення для розв’язання задачі проведення аналізу набору математичних моделей для здійснення найточнішого прогнозування.

3.1.Лінійна регресія

Лінійна регресія — це лінійна модель, наприклад модель, яка передбачає лінійну залежність між вхідними змінними x та єдиною вихідною змінною y. Більш конкретно, що y можна обчислити з лінійної комбінації вхідних змінних x.
Коли є одна вхідна змінна x, метод називається простою лінійною регресією. Коли є кілька вхідних змінних — множинною лінійною регресією [2].

Для підготовки або навчання рівняння лінійної регресії на основі даних можна використовувати різні методи, найпоширеніший з яких називається методом найменших квадратів. Тому прийнято називати підготовлену таким чином модель як “звичайна найменша квадратична лінійна регресія” або просто “регресія найменших квадратів”.

Представлення — це лінійне рівняння, яке поєднує певний набір вхідних значень x, рішенням якого є прогнозований вихід для цього набору вхідних значень y. Таким чином, як вхідні значення x, так і вихідні значення є числовими.
Лінійне рівняння присвоює один коефіцієнт масштабування кожному вхідному значенню чи стовпчику який позначається як β. Також додається один додатковий коефіцієнт, що надає лінії додатковий ступінь свободи (наприклад, рух вгору і вниз по двовимірному графіку) і часто називається коефіцієнтом перехоплення або

зміщення. Наприклад, у простій задачі регресії (один x та один y) модель описується рівнянням (3.1).

$$y_t = \beta_0 + \beta_1 x_t + \varepsilon_t \tag{3.1}$$

де β_0 і β_1 — коефіцієнти регресії; ε — помилка моделі, t – момент часу, яке візуальне представлення якого зображене на рисунку 3.1.

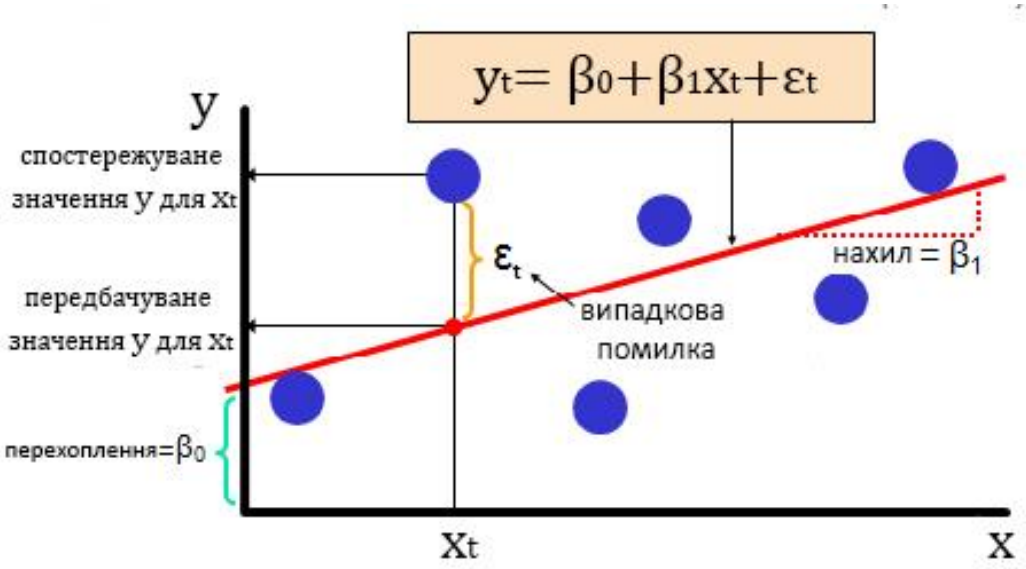


Рисунок 3.1 — Візуальне представлення лінійної регресії

У вищих вимірах, коли існує більше одного входу x, лінію називають площиною або гіперплощиною. Модель прогнозування продемонстроване у рівнянні (3.2).

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \cdots + \beta_n x_{n,t} + \varepsilon \tag{3.2}$$

Коли коефіцієнт стає нульовим, він ефективно знімає вплив вхідної змінної на модель і, отже, з передбачення, зробленого з моделі ($0 * x = 0$). Існують розширення навчання лінійної моделі, які називаються методами регуляризації. Вони прагнуть як мінімізувати суму похибки квадрата моделі на тренувальних даних (використовуючи звичайні найменші квадрати), так і зменшити складність моделі (наприклад, кількість або абсолютний розмір суми всіх коефіцієнтів у моделі)

3.2. Дерево прийняття рішень для регресії (CART)

Дерева рішень, які також сьогодні відомі як дерева класифікації та регресії (CART), були введені Лео Брейманом для позначення алгоритмів дерева рішень. Вони належать до сімейства алгоритмів навчання з учителем, які мають заздалегідь задану цільову змінну, і вони, в основному, використовуються в нелінійному прийнятті рішень з простою лінійною поверхнею рішень.

Одними з найкращих і найбільш застосовуваних методів навчання з учителем є алгоритми на основі дерев. Вони надають можливості прогнозування моделювання з більшою точністю, кращою стабільністю і забезпечують легкість інтерпретації. На відміну від інших алгоритмів навчання з учителем, алгоритм дерева рішень може використовуватися і для вирішення проблем регресії та класифікації. Такі методи, як дерева рішень, випадковий ліс, градієнтне підсилювання широко використовуються у багатьох видах задач Data Science.

Загальний мотив використання дерева рішень полягає у створенні навчальної моделі, яка може використовуватись для прогнозування класу або значення цільових змінних, вивчаючи правила прийняття рішень, виведені з попередніх даних (навчальних даних).

Алгоритм дерева рішень намагається вирішити проблему за допомогою представлення дерева. Кожен внутрішній вузол дерева відповідає атрибуту, а кожному кінцевому вузлу (листя) відповідає мітка класу.

Дерево рішень — структура дерева, схожа на блок-схему, де внутрішній вузол представляє характеристику (або атрибут), гілка представляє правило рішення, а кожен вузол листів представляє результат. Найвищий вузол у дереві рішень відомий як кореневий вузол. Він вчиться розділяти на основі значення атрибута. Він розділяє дерево рекурсивно, що називається рекурсивним поділом.

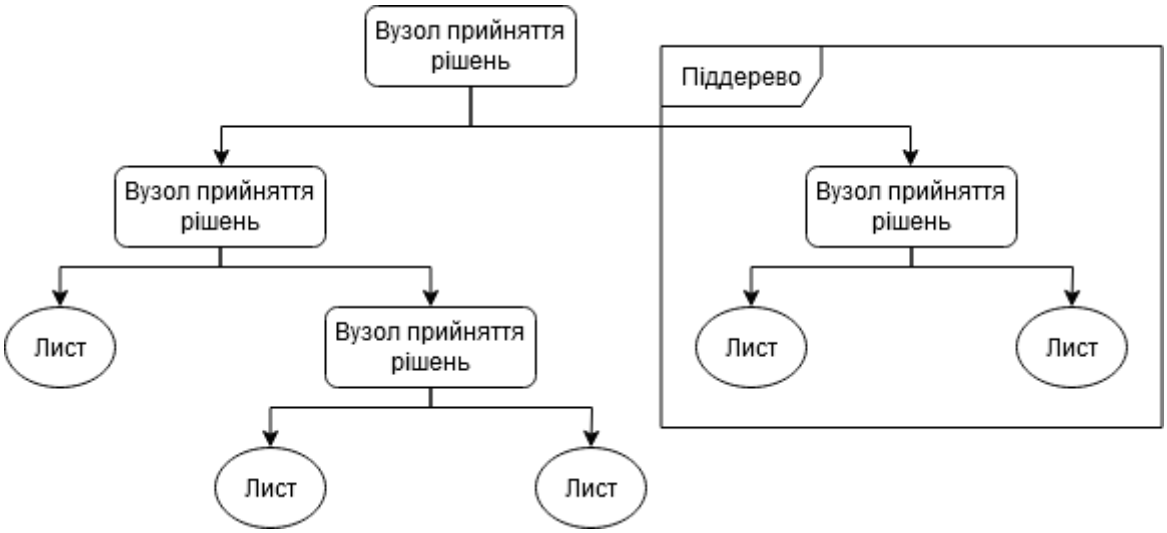


Рисунок 3.2 — Візуальне представлення дерева рішень

Дерево рішень — це алгоритм Machine Learning типу “white box”. Час його навчання швидше порівняно з алгоритмом нейронної мережі [8]. Складність часу дерев рішень — це функція кількості записів і кількості атрибутів у заданих даних. Дерево рішень — це метод без розподілу чи непараметризований метод, який не залежить від припущень щодо розподілу ймовірностей. Дерева рішень можуть обробляти дані високих розмірів з хорошою точністю.

Основна ідея будь-якого алгоритму дерева рішень полягає в наступному:

1. Вибір найкращого атрибуту за допомогою заходів вибору атрибутів (ASM) для розділення записів.

2. Перетворення цього атрибуту у вузол рішення і розбиття набору даних на менші підмножини.

3. Початок побудови дерева, повторення цього процесу рекурсивно для кожного дочірнього елемента , поки одна з умов не буде відповідати:

— Усі кортежі належать до одного і того ж значення атрибута.

— Більше немає атрибутів.

— Більше немає випадків.

Міра вибору атрибутів — евристика для вибору критерію розбиття, який розділяє дані найкращим способом. Він також відомий як правила розбиття, оскільки він допомагає нам визначити точки зупинки для кортежів на заданому вузлі. ASM забезпечує рейтинг кожній функції (або атрибуту), пояснюючи даний набір даних.

Атрибут з найкращою оцінкою буде обраний як атрибут розбиття. У випадку атрибуту з безперервним значенням також необхідно визначити точки розділення для гілок. Найпопулярніші заходи відбору — інформаційний приріст, коефіцієнт підсилення та індекс Джині.

Переваги та недоліки алгоритму дерев рішень.

Переваги:

— Дерева рішень легко пояснити.

— Він дотримується того самого підходу, якого люди дотримуються під час прийняття рішень.

— Інтерпретацію складної моделі дерева рішень можна спростити візуалізаціями.

— Кількість гіпер-параметрів, які потребують налаштування, майже нульова.

Недоліки:

— В дереві рішень велика ймовірність перенавчання.

— Як правило, він дає низьку точність прогнозування для набору даних порівняно з іншими алгоритмами машинного навчання.

— Розрахунки можуть стати складними, коли існує багато міток класу.

Можна підвищити точність цієї моделі завдяки градієнтному бустінгу (GBM). Моделі машинного навчання можуть бути пристосовані до даних окремо або поєднані в ансамблі. Ансамбль — це поєднання простих індивідуальних моделей, які разом створюють більш потужну нову модель.

Прискорення (бустінг) машинного навчання — це метод створення ансамблю моделей. Він починається, після встановлення початкової моделі (наприклад, дерева прийняття рішень або лінійної регресії). Потім будується друга модель, яка фокусується на точному прогнозуванні випадків, коли перша модель працює погано. Очікується, що комбінація цих двох моделей буде кращою, ніж будь-яка модель. Далі, просто треба повторювати цей процес прискорення багато разів. Кожна наступна модель намагається виправити недоліки комбінованого підсиленого ансамблю всіх попередніх моделей.

Градієнтний бустінг спирається на інтуїцію, що найкраща можлива наступна модель у поєднанні з попередніми моделями мінімізує загальну помилку передбачення. Ключова ідея — встановити цільові результати для наступної моделі, щоб мінімізувати помилку. Цільовий результат для кожного випадку в даних залежить від того, наскільки зміна прогнозування цього випадку вплине на загальну помилку прогнозування:

— Якщо невелика зміна прогнозу для випадку спричиняє велике падіння помилки, то наступним цільовим результатом для цього випадку є велике значення. Прогнози щодо нової моделі, наближеної до її цілей, зменшать помилку.

— Якщо невелика зміна прогнозу на випадок не викликає зміни помилки, то наступний цільовий результат для цього випадку дорівнює нулю. Зміна цього прогнозу не зменшує помилку.

Назва “градієнтний бустінг” виникає тому, що цільові результати для кожного випадку встановлюються на основі градієнта помилки щодо прогнозування. Кожна нова модель робить крок у напрямку мінімізації помилки передбачення, у просторі можливих прогнозів для кожного навчального випадку.

3.3. Регресія в нейромережах (ANN)

Штучні нейронні мережі (ANN) складаються з простих елементів, званих нейронами, кожен з яких може приймати прості математичні рішення. Разом нейрони можуть аналізувати складні проблеми, імітувати майже будь-яку функцію, включаючи дуже складні, та давати точні відповіді. Неглибока нейронна мережа має три шари нейронів: вхідний шар, прихований шар та вихідний шар. Глибока нейронна мережа (DNN) має більше ніж один прихований шар, що збільшує складність моделі та може значно підвищити потужність прогнозування.

Розглянемо, наприклад, регресію або проблему класифікації. В обох випадках існують деякі вхідні параметри, позначені $(i_1, i_2, \dots, i_n)$, і потрібно знайти функцію

цих входів, яка досить добре пояснює спостережувані відповідні виходи, позначені $(o_1, o_2, \dots, o_m)$. Іншими словами, як у випадку прикладу лінійної регресії, шукана функція f така, що $(o_1, o_2, \dots, o_m)$ добре апроксимується $f(i_1, i_2, \dots, i_n)$. Ідея моделювання нейронних мереж полягає у тому, щоб забути ідею встановити легко параметризовану функцію, головним чином “сформовану” людиною та відрегульовану машиною (через ці кілька параметрів, як у прикладі лінійної регресії), але замість цього треба встановити сильно параметризовану функцію, дуже гнучку, яка апріорі не має особливого сенсу для людини, але яка буде зручно сформована під час фази навчання. Спробуємо проілюструвати це простою нейронною мережею.

Отже, треба навчити функцію f таку, що $f(i_1, i_2)$ — хороший оцінювач o_1. Тоді можна було б запропонувати наступну першу модель (3.3)

$$f_1(i_1, i_2) = w_{11} i_1 + w_{12} i_2 + \mathrm{c} \tag{3.3}$$

де w_{11} і w_{12} — це лише ваги / коефіцієнти.

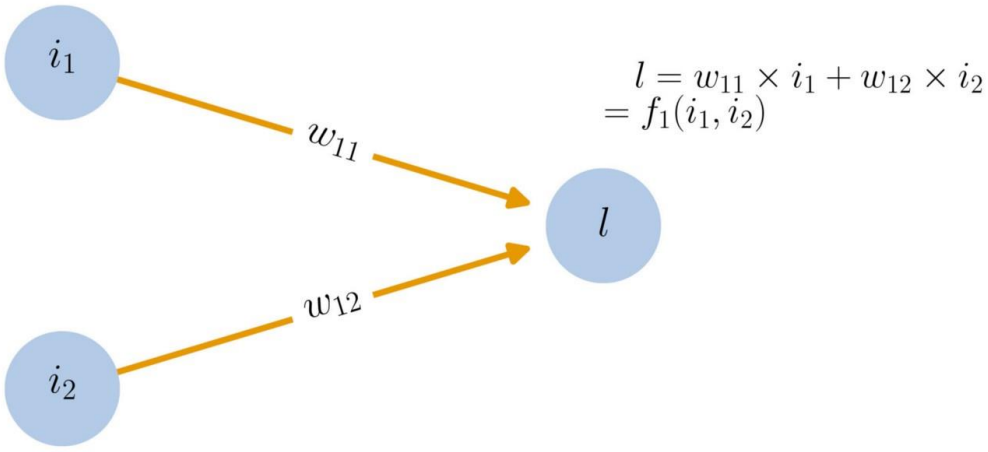


Рисунок 3.3 — Лінійна регресійна модель для декількох входів

Модель яка зображена на рисунку 3.3 — є лінійною регресійною моделлю для декількох входів (також її називають багатолінійною регресією). У цьому випадку модель легко зрозуміти і підігнати, але є великий недолік: в ній немає нелінійності Щоб ввести нелінійність, треба застосувати невелику модифікацію до попередньої моделі, після чого отримаємо рівняння (3.4).

$$f_1(i_1, i_2) = a(w_{11}i_1 + w_{12}i_2 + \text{c}) \quad (3.4)$$

де a — функція під назвою “функція активації”, яка є нелінійною. Тоді можна помітити, що $w_{11}i_1 + w_{12}i_2$ все ще лінійна, але, оскільки це значення стає таким, що воно проходить через нелінійну функцію, то загальний результат стає вже не лінійним і, таким чином, ця модель ближче до нашого припущення, ніж попередня модель, вона зображена на рисунку 3.4.

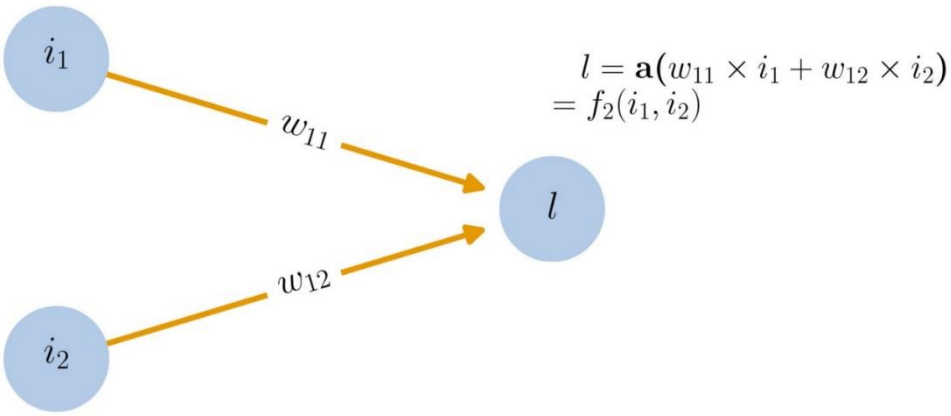


Рисунок 3.4 — Модифікована лінійна регресійна модель з функцією активації

Якщо вибрати в якості функції активації так звану “логістичну функцію”, то таким чином визначається модель під назвою “логістична регресія”, яка може відповідати, наприклад, деяким задачам бінарної класифікації (справді логістична функція видає число від 0 до 1, яке можна розглядати як ймовірність перебування в одному з двох класів).

Однак, навіть якщо вона краще, ніж полілінійна модель, ця модель все ще надто проста і не може впоратися з припущеною базовою складністю взаємозв’язку між входами та результатами. Можна зробити крок далі і збагатити модель наступним чином. Спершу можна було б врахувати, що величина $a(w_{11}i_1 + w_{12}i_2)$ вже не є кінцевим результатом, а натомість є новою проміжною ознакою функції, яка називається l_1. По-друге, можна було б врахувати, що будується кілька (3 у поточному прикладі) таких об’єктів одним і тим самим способом, але, можливо, з різною вагою та різними функціями активації: $l_1 = a_{11}(w_{11}i_1 + w_{12}i_2)$, $l_2 = a_{12}(w_{21}i_1 + w_{22}i_2)$ та $l_3 = a_{13}(w_{31}i_1 + w_{32}i_2)$, де a є лише функціями активації, а w

— вагами. Нарешті, можна вважати, що наш кінцевий результат будується на основі цих проміжних об’єктів з тим же “шаблоном”: $a_2(v_1 l_1 + v_2 l_2 + v_3 l_3)$. Якщо об’єднати всі частини, то отримаємо рівняння (3.5).

$$f_3(i_1, i_2) = a_2(v_1 l_1 + v_2 l_2 + v_3 l_3) = a_2(v_1 a_{11}(w_{11} l_1 + w_{12} l_2) + v_2 a_{12}(w_{21} i_1 + w_{22} i_2) + v_3 a_{13}(w_{31} i_1 + w_{32} i_2)) \quad (3.5)$$

де a — це нелінійні функції активації, а w і v — ваги. На рисунку 3.5 проілюстроване мережеве графічне представлення цієї і для попередніх моделей.

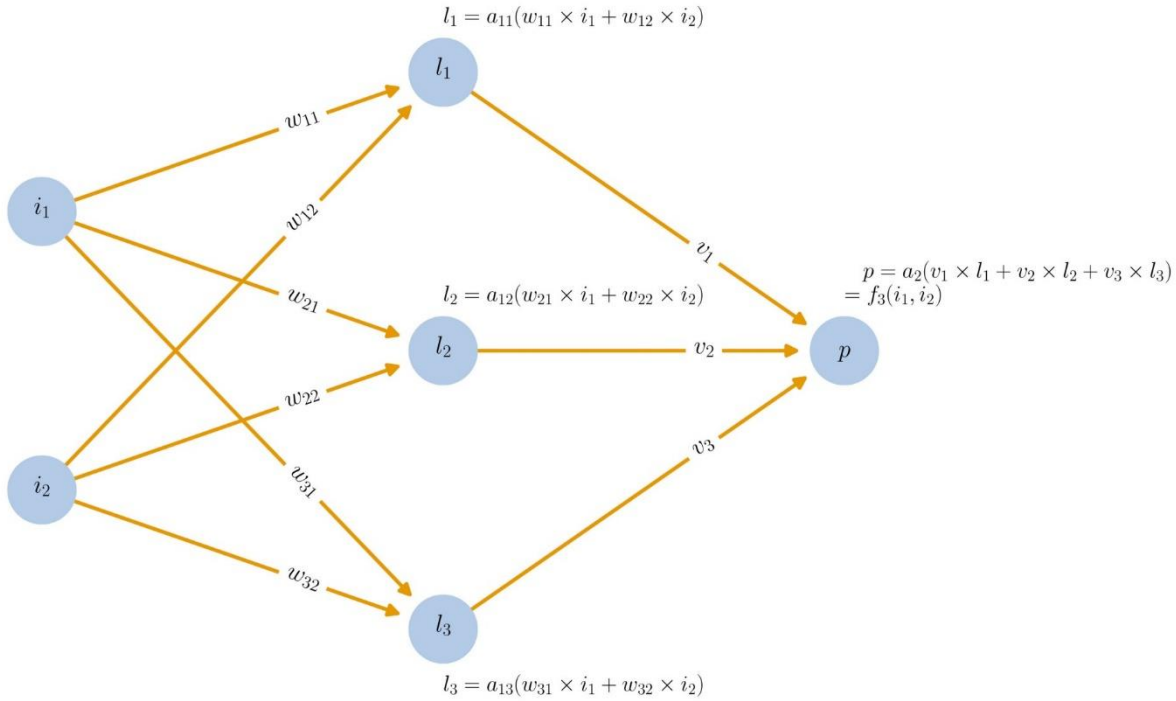


Рисунок 3.5 — Графічне представлення ускладненої моделі

Ця остання модель є базовою нейронною мережею прямого розповсюдження, з 2 входами (i_1 та i_2), 1 прихованим шаром з 3 прихованими нейронами (виходи яких l_1, l_2 та l_3) та 1 кінцевим результатом. Можна було б вирішити додати ще один проміжний “прихований” шар між l і кінцевим виходом точно так же, як просто додаються ці l між входом і виходом: тоді можна було б мати нейронну мережу з 2 прихованими шарами. Або можна було б вирішити залишитися з одним прихованим шаром, але мати в ньому більше нейронів (5 замість 3, наприклад). Отже, є різні способи ускладнення моделі, завдяки якій кількість ваг зростатиме. Кількість

прихованих шарів, кількість нейронів у кожному шарі, знання ваг та характер активаційних функцій визначають нейронну мережу і, таким чином, “шаблон” функції, описуваною цією функцією.

Нарешті, можна зазначити кілька важливих речей. По-перше, на шляху математичного опису цих мереж можна записати всі попередні рівняння з матричними позначеннями: це робить їх набагато більш легшими для читання, для великих архітектур. По-друге, всі нейронні мережі не відповідають шаблону, описаному вище: існують різні види архітектури, але нейронна мережа, що подається, є дійсно першою базовою архітектурою, яка зрозуміла. По-третє, як тільки буде визначено, модель все-таки повинна бути встановлена (ваги повинні бути відрегульовані, виходячи з даних, щоб мінімізувати деяку функцію помилок, як і у випадку лінійної регресії), і це справді складна задача оптимізації для завершення.

3.4. Алгоритм ARIMAX

Моделі часових рядів використовують минулі рухи змінних для прогнозування їх майбутніх значень. На відміну від структурних моделей, які пов’язують змінну, яку треба прогнозувати, з набором інших змінних, модель часових рядів не базується на економічній теорії. Однак, з точки зору прогнозування, надійність оцінюваного рівняння повинна базуватися на результатах поза вибіркою. Модель часових рядів може здебільшого давати досить точні прогнози, особливо у випадку наявності багатовимірних зв’язків між змінними.

Модель часових рядів із застосуванням підходу Box-Jenkins була запропонована Джорджом Боксом та Гвілімом Дженкінсоном у 1970-му році. Цей підхід широко застосовується через його ефективність та простоту. Більшість часових рядів можна описати за допомогою моделі Autoregressive Moving Average (ARMA). Стаціонарний ряд Y_t позначається як $ARMA\ (p, q)$, якщо рівняння (3.5) виконується.

$$Y_t - \varphi_1 Y_t - 1 - \cdots - \varphi_p Y_t - \mathrm{p} = \varepsilon_t + \theta_1 \varepsilon_t - 1 + \cdots + \theta_q \varepsilon_t - \mathrm{q} \qquad (3.6)$$

де ε_t — білий шум і немає загального фактора між авторегресивним многочленом $(1 - \varphi_1 \mathrm{L}_1 - \varphi_2 \mathrm{L}_2 - \cdots - \varphi_p \mathrm{L}_p)$ і ковзним середнім многочленом $(1 + \theta_1 \mathrm{L}_1 + \cdots + \theta_q \mathrm{L}_q)$, де L – оператор відставання [6]. Також ці многочлени можуть бути представлені як $\varphi(L)$ та $\theta(L)$ відповідно.

Якщо ряд є різнице-стійким, реалізується модель (ARIMA). ARIMA розшифровується як моделі Авторегресійного Інтегрованого Ковзного Середнього. Одновимірна (одновекторна) ARIMA — це метод прогнозування, який проектує майбутні значення ряду цілком на основі власної інерції. Основне його застосування — у галузі короткострокового прогнозування, що вимагає не менше 40 історичних точок даних. Найкраще це працює, коли дані демонструють стабільний або послідовний паттерн у часі з мінімальною кількістю викидів. ARIMA, як правило, перевершує експоненціальні методи згладжування, коли дані досить довгі і кореляція між минулими спостереженнями стабільна.

Першим кроком у застосуванні методології ARIMA є перевірка її стаціонарності. “Стаціонарність” передбачає, що ряд з часом залишається на досить постійному рівні. Якщо існує тенденція, то дані не є стаціонарними. Дані також повинні показувати постійну дисперсію в його коливаннях у часі. Це легко помітити в ряді, який сильно сезонний і зростає швидшими темпами. У такому випадку підйоми та падіння сезонності з часом стануть більш різкими. Без виконання цих умов стаціонарності багато обчислень, пов’язаних із процесом, неможливо обчислити.

Якщо візуалізація даних вказує на нестаціонарність, то слід “диференціювати” ряд. Диференціація — це спосіб перетворення нестаціонарного ряду в стаціонарний. Це робиться шляхом віднімання спостереження поточного періоду від попереднього. Якщо це перетворення застосовується до ряду лише один раз, вважається, що дані були “вперше продиференційованими”. Цей процес, по суті, усуває тенденцію, якщо даний ряд росте постійними темпами. Якщо він зростає зі зростаючими темпами, можна застосувати ту саму процедуру та знову продиференціювати дані. Ваші дані потім будуть “вдруге продиференційованими”.

“Автокореляції” — це числові значення, які вказують на те, як ряд даних пов’язаний із самим собою з часом. Точніше, він вимірює, наскільки сильно значення даних за певну кількість періодів один від одного співвідносяться один з одним у часі. Кількість періодів один від одного зазвичай називають “лагом”. Наприклад, автокореляція з лагом 1 вимірює, як значення з різницею в 1 період корелюють один з одним протягом усього ряду. Автокореляція з лагом 2 вимірює, як дані між двома періодами співвідносяться протягом усього ряду. Автокореляція може коливатися від +1 до -1. Значення, близьке до +1, вказує на високу позитивну кореляцію, тоді як значення, близьке до -1, означає високу негативну кореляцію. Ці показники найчастіше оцінюються за допомогою графіків, які називаються “корелограмами”. Корелограма будує графіки значень автокореляції для даного ряду з різним лагом. Це називається “функцією автокореляції” і є дуже важливим у методі ARIMA.

Методологія ARIMA намагається описати рухи в стаціонарному часовому ряді в залежності від так званих параметрів “авторегресії та ковзного середнього”. Вони називаються AP параметрами (авторегресивними) та параметрами MA (ковзні середні значення). Модель AR з лише 1 параметром може бути записана як рівняння (3.7).

$$\mathrm{X(t)} = \mathrm{A(1)}\ \mathrm{X(t-1)} + \mathrm{E(t)} \tag{3.7}$$

де $X(t)$ — часовий ряд, що досліджується, $A(1)$ — автоматичний параметр порядку 1, $X(t-1)$ — часовий ряд, що відставав 1 період та $E(t)$ — термін помилки моделі. Це просто означає, що будь—яке задане значення $X(t)$ може бути пояснено деякою функцією від його попереднього значення, $X(t-1)$, плюс деякою випадковою помилкою, $E(t)$. Якщо розрахункове значення $A(1)$ становило 0,30, то поточне значення ряду було б пов’язане з 30% від його значення 1 період тому. Звичайно, ряд може бути пов’язаний з більш ніж одним минулим значенням. Наприклад, рівняння (3.8).

$$X(t) = A(1)X(t-1) + A(2)\,X(t-2)\; +\; E(t) \tag{3.8}$$

Це вказує на те, що поточне значення ряду є комбінацією двох попередніх значень, $X(t-1)$ і $X(t-2)$, плюс деяка випадкова помилка $E(t)$. Ця модель зараз є авторегресивною моделлю другого порядку.

Наступний тип моделі Box-Jenkins називається моделлю “ковзного середнього”. Хоча ці моделі дуже схожі на модель AR, їх концепція зовсім інша. Параметри ковзного середнього зв’язують те, що відбувається в періоді t, лише з випадковими помилками, які мали місце в минулих періодах часу, тобто $E(t-1)$, $E(t-2)$ тощо, а не $X(t-1)$, $X(t-2)$, $X(t-3)$, як в авторегресивних підходах. Модель ковзного середнього з одним терміном MA може бути записана як рівняння (3.9).

$$X(t) = -B(1)\,E(t-1) + E(t) \tag{3.9}$$

Термін $B(1)$ називається MA порядку 1. Від’ємний знак перед параметром використовується лише для конвенції і зазвичай роздруковується автоматично більшістю комп’ютерних програм. Вищенаведена модель просто говорить, що будь—яке задане значення $X(t)$ безпосередньо пов’язане лише з випадковою помилкою в попередньому періоді, $E(t-1)$, і з поточним членом помилки, $E(t)$. Як і у випадку з авторегресивними моделями, моделі ковзних середніх можна поширити на структури вищого порядку, що охоплюють різні комбінації та довжини ковзного середнього.

Методика ARIMA також дозволяє будувати моделі, що включають як авторегресивні, так і ковзні середні разом. Ці моделі часто називають “змішаними моделями”. Хоча це робить інструмент прогнозування більш складнішим, структура може дійсно краще імітувати ряд та продукувати більш точний прогноз. З чистої моделі випливає, що структура складається лише з параметрів AR або MA — не з обох.

Моделі, розроблені таким підходом, зазвичай називаються моделями ARIMA, оскільки вони використовують комбінацію авторегресивних (AR), інтеграційних (I)

— відноситься до зворотного процесу диференціювання для отримання прогнозу та операцій ковзного середнього (MA). Модель ARIMA зазвичай називається $ARIMA\ (p, d, q)$. Це представляє порядок авторегресивних компонентів (p), кількість операторів диференціювання d та найвищий порядок ковзного середнього. Наприклад, $ARIMA\ (2,1,1)$ означає авторегресивну модель другого порядку з компонентом ковзного середнього першого порядку, ряд якої був диференційований один раз, для перетворення у стаціонарний.

Моделі Arimax – це сезонна модифікація Arima. Ця модифікація проводиться шляхом додавання змінних предиктора. Ефекти календаря варіацій є однією із змінних прогнозів, які часто використовуються при моделюванні. Позначимо модель Арімакс з варіантами ефектів календаря

Емпіричні дослідження з макроекономічного прогнозування, такі як Stock and Watson (1999), показали, що включення випереджуючих показників у модель покращує ефективність прогнозування. Hamiltion an Perez-Quiros (1996) припустив, що складений випереджуючий показник (CLI — Composite Leading Indicator) — хороший предиктор для ВВП, особливо у випадку переламного періоду в економіці. Модель ARIMA розширена пояснювальною змінною X — називається $ARIMAX\ (p, d, q)$. Зокрема, $ARIMAX\ (p, d, q)$ може бути представлений як (3.10).

$$\varphi(\mathrm{L})(1 - \mathrm{L})\ \mathrm{dY}_t = \Theta(\mathrm{L})\mathrm{X}_t + \Theta(\mathrm{L})\varepsilon_t \qquad (3.10)$$

де X_t є CLI торгового партнера.

Основна проблема в класичній моделі Box-Jenkins яку намагаються вирішити, яку специфікацію ARIMA використовувати, наприклад, скільки AR та / або MA параметрів, які потрібно включити. Саме цьому значна частина Box-Jenkings була присвячена “процесу ідентифікації”. Це залежало від графічного та чисельного оцінювання автокореляції вибірки та часткової функції автокореляції.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для створення даного програмного продукту використовувався фреймворк Flask. Він був вибраний завдяки надання можливості швидкої розробки додатку, наявності широкого спектру розширень, та легкої інтеграції з проектами Data Mining, розробленими на мові програмування Python.

4.1. Фреймворк Flask

Flask — це легкий фреймворк, написаний мовою програмування Python. Він відрізняється від інших фреймворків, оскільки дозволяє розробникам зайняти місце “водія” та мати повний творчий контроль над своїми програмами. В інших фреймворках, коли потрібно відхилитися від шляху визначеного їх розробниками, наприклад, використовувати інший механізм баз даних або, можливо, інший метод аутентифікації користувачів, виникає дуже багато проблем. Flask принципово відрізняється від їх усіх. За допомогою Flask можна вибрати компоненти для розроблюваного додатку або написати свій. Ключ до цієї свободи полягає в тому, що Flask розробляли з самого початку для розширення. Він оснащений надійним ядром, що включає в себе основний функціонал, який потрібні всім веб-додаткам, і очікує, що решту нададуть деякі з багатьох розширень сторонніх розробників в екосистемі, або сам користувач. Цей фреймворк базується на двох бібліотеках: Werkzeug та Jinja 2.

Werkzeug — це комплексна бібліотека веб-додатків WSGI. WSGI означає “Web Server Gateway Interface”. Це специфікація, яка описує, як веб-сервер спілкується з веб-додатками та як веб-додатки можуть бути пов'язані разом для обробки одного

запиту. Werkzeug створювалася як проста колекція різних утиліт для додатків WSGI і стала однією з найдосконаліших бібліотек утиліт WSGI.

Jinja2 — це шаблонна мова для Python з потужним набором інструментів та інтуїтивним дизайном. Jinja2 дозволяє використовувати логіку шаблону, додає багаторазові макро-вирази та основні математичні функції. Він забезпечує високу продуктивність і сумісний з широким спектром версій Python. Jinja2 використовує Unicode у внутрішній реалізації і надає Python-подібні вирази. Це гнучкий текстовий механізм шаблонів і може генерувати будь-який вихідний код або код розмітки.

Flask — це невеликий фреймворк за більшістю стандартів, достатньо малий, щоб його можна назвати “мікро-фреймворк”. У Flask немає вбудованої підтримки для доступу до баз даних, перевірки веб-форм, аутентифікації користувачів або інших завдань високого рівня. Ці та багато інших ключових служб, які потребують більшості веб-додатків, доступні через розширення, що інтегруються з основними пакетами. Як розробник, ви маєте змогу вибирати розширення, які найкраще працюють для вашого проекту, або навіть писати власні.

Хоча наявність невеликих веб-додатків, що зберігаються в одному сценарії, може бути дуже зручною, але такий підхід ускладнює масштабування. Оскільки складність додатка зростає, робота з одним великим вихідним файлом стає проблематичною. На відміну від більшості інших веб-фреймворків, Flask не нав'язує конкретної організації для великих проектів; спосіб структурування додатку повністю залишається розробнику.

4.2. Опис роботи додатку

4.2.1. Попередня обробка датасету

Перший етап даної роботи являє собою важливу техніку Data Mining, яка в основному стосується очищення та перетворення вихідних даних у корисний і зрозумілий формат [4, 10]. Raw data (необроблені дані) часто є неповними, непослідовними і можуть не мати певної поведінки, і, ймовірно, містять багато помилок, застосування алгоритмів обробки даних — призведе до зменшення неточності. Попередня обробка даних — це перевірений метод вирішення таких питань, це процес виявлення та виправлення (або вилучення) пошкоджених або неточних записів із набору записів таблиці, або посилання на визначення неповних, невірних, неточних або невідповідних частин даних, а потім заміни, зміни або видалення “брудних” чи “грубих” даних.

У програмі попередньої обробки даних в основному є три важливі підтеми:

- Очищення даних.
- Перетворення даних.
- Зменшення даних.

Очищення даних — це процес очищення необроблених даних шляхом обробки невідповідних та відсутніх кортежів. Під час роботи з проектами машинного навчання набори даних можуть бути не ідеальними, вони можуть мати багато домішок, шумних значень і в більшості разів фактичні дані можуть бути відсутніми.

Перетворення даних робиться для того, щоб перетворити дані у відповідні форми, придатні для видобутку інформації. Це включає нормалізацію, дискретизацію та концепцію генерації ієрархій [3]. Нормалізація — робиться для того, щоб масштабувати значення даних у визначеному діапазоні (від -1,0 до 1,0 або 0,0 до 1,0). Вибір атрибутів — у цій стратегії нові атрибути будуються із заданого набору атрибутів, щоб допомогти процесу видобутку інформації. Дискретизація – це

робиться для того, щоб замінити необроблені значення числового атрибута інтервальними рівнями або понятійними рівнями. Концепція генерації ієрархій — тут атрибути перетворюються на вищий рівень в ієрархії. Наприклад, атрибут “місто” можна перетворити на “країну”.

Оскільки Data Mining — це техніка, яка використовується для обробки величезної кількості даних. Працюючи з такими обсягами даних — аналіз ускладнюється. Для того, щоб позбутися цього, можна використовувати техніку зменшення даних. Вона спрямована на підвищення ефективності зберігання та зниження витрат на зберігання та аналіз даних.

Кроки до зменшення даних:

— Агрегація кубів даних — застосовується до даних для побудови куба даних.

— Вибір підмножини атрибутів — використання високоактуальних атрибутів, та ігнорування всіх інших. Для здійснення вибору атрибута можна використовувати рівень значущості та пі—значення атрибута. Атрибут, який має р-значення, більший за рівень значущості, може бути відкинутий.

— Зменшення чисельності — дає змогу зберігати модель даних замість цілих даних, наприклад: Регресійні моделі.

— Зменшення розмірності — зменшує розмір даних за допомогою механізмів кодування. Вони можуть бути “витратними” або “без втрат”. Якщо після реконструкції зі стислих даних можуть бути отримані вихідні дані, таке зменшення називається зменшенням без втрат, інакше воно називається зменшенням із втратами.

У даній роботі використовується датасет під назвою “Us Accidents”, коротка інформація про нього зображена на рисунку 4.1.

```
RangeIndex: 2974335 entries, 0 to 2974334
Data columns (total 49 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   ID                     2974335 non-null  object
 1   Source                 2974335 non-null  object
 2   TMC                    2246264 non-null  float64
 3   Severity               2974335 non-null  int64
 4   Start_Time             2974335 non-null  object
 5   End_Time               2974335 non-null  object
 6   Start_Lat              2974335 non-null  float64
 7   Start_Lng              2974335 non-null  float64
 8   End_Lat                728071 non-null   float64
 9   End_Lng                728071 non-null   float64
 10  Distance(mi)           2974335 non-null  float64
 11  Description            2974334 non-null  object
 12  Number                 1056730 non-null  float64
 13  Street                 2974335 non-null  object
 14  Side                   2974335 non-null  object
 15  City                   2974252 non-null  object
 16  County                 2974335 non-null  object
 17  State                  2974335 non-null  object
 18  Zipcode                2973455 non-null  object
 19  Country                2974335 non-null  object
 20  Timezone               2971172 non-null  object
 21  Airport_Code           2968644 non-null  object
 22  Weather_Timestamp      2937630 non-null  object
 23  Temperature(F)         2918272 non-null  float64
 24  Wind_Chill(F)          1121712 non-null  float64
 25  Humidity(%)            2915162 non-null  float64
 26  Pressure(in)           2926193 non-null  float64
 27  Visibility(mi)         2908644 non-null  float64
 28  Wind_Direction         2929234 non-null  object
 29  Wind_Speed(mph)        2533495 non-null  float64
 30  Precipitation(in)      975977 non-null   float64
 31  Weather_Condition      2908403 non-null  object
 32  Amenity                2974335 non-null  bool
 33  Bump                   2974335 non-null  bool
 34  Crossing               2974335 non-null  bool
 35  Give_Way               2974335 non-null  bool
 36  Junction               2974335 non-null  bool
 37  No_Exit                2974335 non-null  bool
 38  Railway                2974335 non-null  bool
 39  Roundabout             2974335 non-null  bool
 40  Station                2974335 non-null  bool
 41  Stop                   2974335 non-null  bool
 42  Traffic_Calming        2974335 non-null  bool
 43  Traffic_Signal         2974335 non-null  bool
 44  Turning_Loop           2974335 non-null  bool
 45  Sunrise_Sunset         2974242 non-null  object
 46  Civil_Twilight         2974242 non-null  object
 47  Nautical_Twilight      2974242 non-null  object
 48  Astronomical_Twilight  2974242 non-null  object
dtypes: bool(13), float64(14), int64(1), object(21)
memory usage: 853.8+ MB
```

Рисунок 4.1 — Загальна інформація про вміст датасету “Us Accidents”

Для реалізації першого етапу, спочатку було проаналізовано відсотковий вміст пропущених значень для кожної характеристики датасету, та виявлено поля які є зайвими і мало інформативними, завдяки візуалізації результатів цього аналізу зображеними на рисунку 4.2.

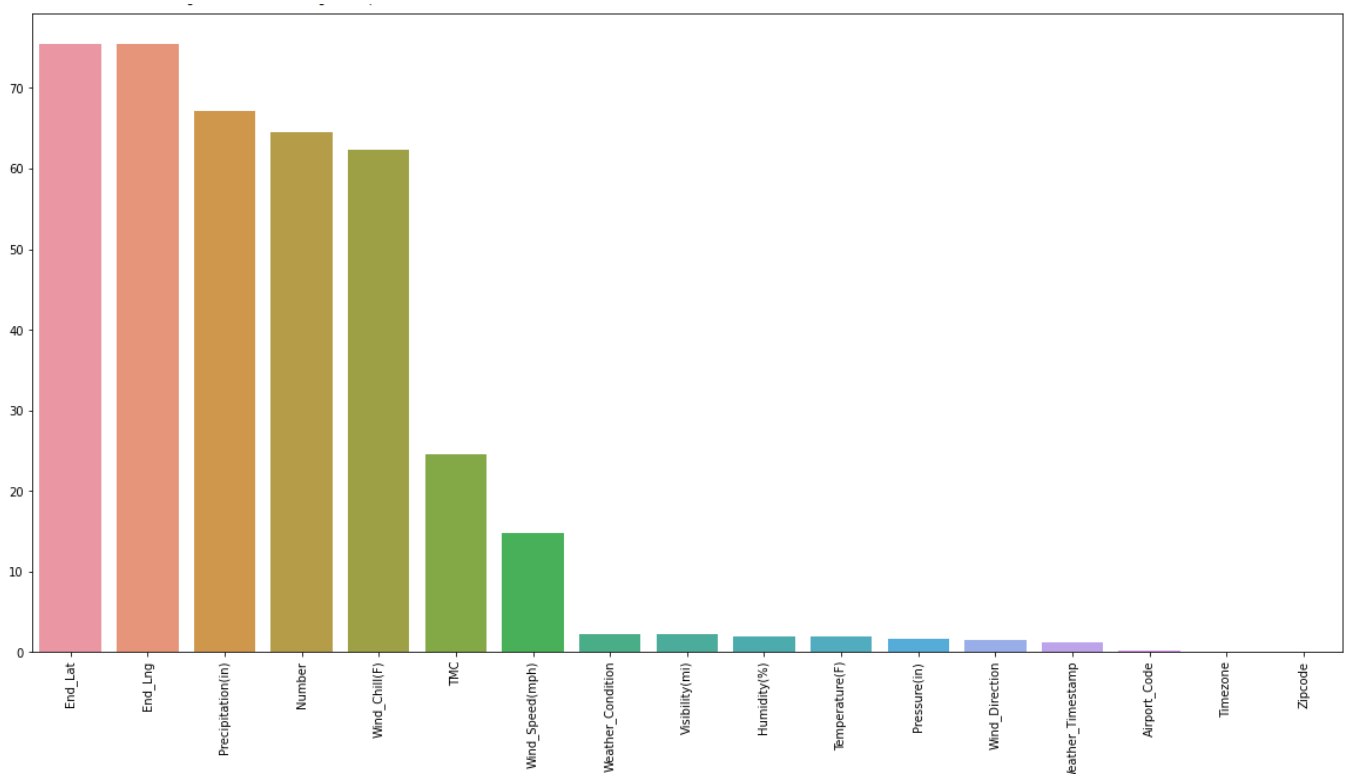


Рисунок 4.2 — Відсотковий вміст пропущених значень у датасеті

Отримавши результати аналізу, було вилучено поля які мали найбільшу кількість пропущених значень. Наступним етапом була конвертація типів за замовчуванням, у відповідні до значень цих полів типи. Наприклад, у характеристики “Severity” за замовчуванням був заданий тип “int64”, хоча це поле — ціле число з діапазоном значень лише від 1 до 4, тому для оптимізації та полегшенню роботи над датасетом, тип цього поля був змінений на “int16”. Також, задля прискорення аналізу було змінено міри деяких стовпчиків на європейські. Одним з них є стовпчик під назвою “Temperature(F)”, який містить інформацію про температуру повітря, яка була під час аварії. Це поле вимірюється у Фарингейтах (F), які використовуються лише у декількох країнах світу, як міра температури. Для досягнення легшого сприйняття більшою кількістю людей, це поле було конвертовано у значення температури, у градусах Цельсія (C). Для знаходження ключових полів для аналізу було використано алгоритм “Дерева рішень для регресії” та “бустінг” для нього, візуалізація результатів якого, зображена на рисунку 4.3. Після всіх перетворень та зменшень кількості даних датасет став важити майже у два рази менше та став більш зрозумілим, це зображено на рисунку 4.4, чим була доведена практична значущість цього етапу.

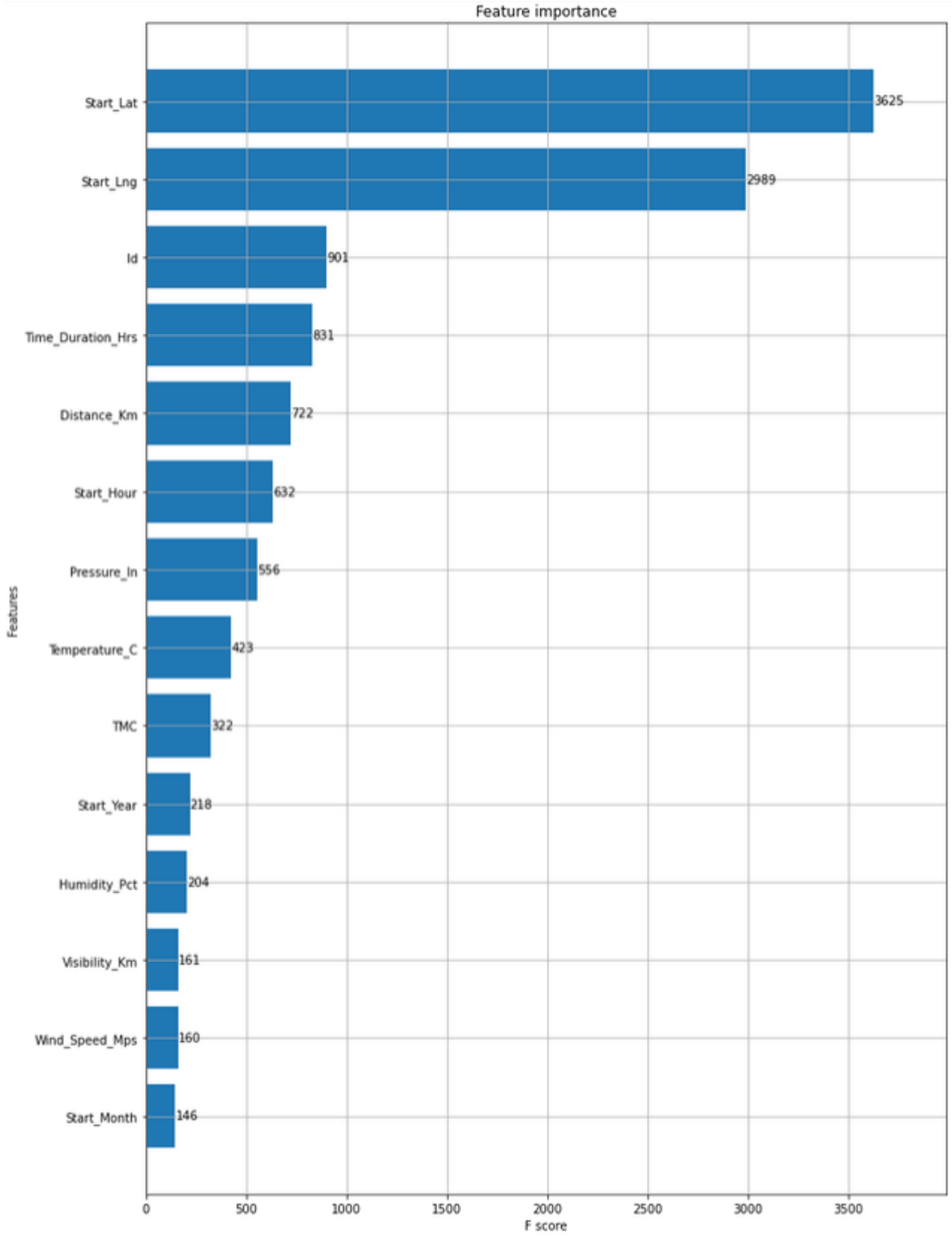


Рисунок 4.3 — Візуалізація значущості критеріїв датасету

```
RangeIndex: 2974335 entries, 0 to 2974334
Data columns (total 40 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   TMC                 2246264 non-null  float64
 1   Severity            2974335 non-null  int16
 2   Start_Time          2974335 non-null  datetime64[ns]
 3   End_Time            2974335 non-null  datetime64[ns]
 4   Start_Lat           2974335 non-null  float16
 5   Start_Lng           2974335 non-null  float16
 6   Distance_Km         2974335 non-null  float64
 7   Description         2974334 non-null  string
 8   Side                2974335 non-null  string
 9   City                2974252 non-null  string
 10  County              2974335 non-null  category
 11  State               2974335 non-null  category
 12  Timezone            2971172 non-null  string
 13  Weather_Timestamp   2937630 non-null  datetime64[ns]
 14  Temperature_C       2918272 non-null  float64
 15  Humidity_Pct        2915162 non-null  float16
 16  Pressure_In         2926193 non-null  float16
 17  Visibility_Km       2908644 non-null  float64
 18  Wind_Direction      2929234 non-null  string
 19  Wind_Speed_Mps      2533495 non-null  float64
 20  Weather_Condition   2908403 non-null  category
 21  Amenity             2974335 non-null  bool
 22  Bump                2974335 non-null  bool
 23  Crossing            2974335 non-null  bool
 24  Give_Way            2974335 non-null  bool
 25  Junction            2974335 non-null  bool
 26  No_Exit             2974335 non-null  bool
 27  Railway             2974335 non-null  bool
 28  Roundabout          2974335 non-null  bool
 29  Station             2974335 non-null  bool
 30  Stop                2974335 non-null  bool
 31  Traffic_Calming     2974335 non-null  bool
 32  Traffic_Signal      2974335 non-null  bool
 33  Turning_Loop        2974335 non-null  bool
 34  Sunrise_Sunset      2974242 non-null  string
 35  Id                  2974335 non-null  int64
 36  Time_Duration_Hrs   2974335 non-null  float16
 37  Start_Year          2974335 non-null  int16
 38  Start_Month         2974335 non-null  int16
 39  Start_Hour          2974335 non-null  int16
dtypes: bool(13), category(3), datetime64[ns](3), float16(5), float64(5), int16(4), int64(1), string(6)
memory usage: 439.8 MB
```

Рисунок 4.4 — Загальна інформація про вміст “Us Accidents” після першого етапу

Отже, завдяки даному етапу було отримано датасет який має меншу вагу, конкретизовані за типом поля та змінені міри деяких даних задля підвищення значущості результатів, та прискорення аналізу.

4.2.2. Візуалізація значущих критеріїв датасету

Наступний етап — візуалізація даних, вона забезпечує потужний механізм допомоги користувачу як під час попередньої обробки даних, так і при фактичній обробці даних. Завдяки візуалізації вихідних даних, користувач може переглядати, щоб отримати “відчуття” властивостей цих даних. Наприклад, великі зразки можна візуалізувати та проаналізувати. Зокрема, візуалізація може бути використана для виявлення зовнішньої форми, яка висвітлює незвичну поведінку в даних, тобто випадки даних, які не відповідають загальній поведінці або моделі. Крім того, користувачеві допомагають у виборі відповідних даних через візуальний інтерфейс. Перетворення даних є важливим кроком попередньої обробки даних. Під час перетворення даних візуалізація даних може допомогти користувачеві забезпечити правильність перетворення. Тобто, користувач може визначити, чи є два представлення даних (оригінальні та перетворені) рівнозначними. Візуалізація може також використовуватися для допомоги користувачам при інтеграції джерел даних, допомагаючи їм бачити взаємозв’язки в різних форматах.

Методи візуалізації даних класифікуються стосовно трьох аспектів. По-перше, їх спрямованість, тобто символічне проти геометричного; по-друге, їх стимул (2D та 3D); і нарешті, їх відображення (статичне або динамічне). Крім того, дані в сховищі даних можуть розглядатися на різних рівнях деталізації або абстрагування, або як різні комбінації атрибутів або розмірів. Дані можуть бути представлені у різних візуальних форматах, включаючи коробкові графіки, точкові діаграми, 3D-кубів, діаграми розподілу даних, криві, візуалізація об’єму, поверхонь або графіки зав’язків.

Для початку етапу візуалізації, були переглянуті основні статистичні дані про датасет, а саме середнє значення, максимальне, мінімальне, медіану та інші, для кожного числового стовпчика, які зображені на рисунку 4.5.

	TMC	Severity	Start_Lat	Start_Lng	End_Lat	End_Lng	Distance_Km	Temperature_C	Wind_Chill_C	Humidity_Pct	Pressure_In	Visibility_Km	Wind_Speed_Mps	Precipitation_In
count	2.2463e+06	2.9743e+06	2.9743e+06	2.9743e+06	2.9743e+06	2.9743e+06	2.9743e+06	2.9183e+06	1.1217e+06	2915162.0	2.9262e+06	2.9086e+06	2.5335e+06	975977.0
mean	2.0783e+02	2.3602e+00	NaN	NaN	NaN	NaN	4.5957e-01	1.6862e+01	1.0737e+01	NaN	NaN	1.4727e+01	3.7097e+00	0.0
std	2.0330e+01	5.4147e-01	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	2.4919e+00	1.0438e+01	1.3995e+01	0.0	0.0000e+00	4.6544e+00	2.2971e+00	0.0
min	2.0000e+02	1.0000e+00	2.4562e+01	-1.2462e+02	2.4562e+01	-1.2462e+02	0.0000e+00	-6.1007e+01	-5.4375e+01	1.0	0.0000e+00	0.0000e+00	0.0000e+00	0.0
25%	2.0100e+02	2.0000e+00	3.3562e+01	-1.1731e+02	3.3562e+01	-1.1731e+02	0.0000e+00	1.0000e+01	0.0000e+00	49.0	2.9812e+01	1.6093e+01	2.0571e+00	0.0
50%	2.0100e+02	2.0000e+00	3.5844e+01	-9.0250e+01	3.5844e+01	-9.0250e+01	0.0000e+00	1.7986e+01	1.2222e+01	67.0	2.9984e+01	1.6093e+01	3.1293e+00	0.0
75%	2.0100e+02	3.0000e+00	4.0375e+01	-8.0938e+01	4.0375e+01	-8.0938e+01	1.6097e-02	2.4444e+01	2.2778e+01	84.0	3.0109e+01	1.6093e+01	4.6485e+00	0.0
max	4.0600e+02	4.0000e+00	4.9000e+01	-6.7125e+01	4.9000e+01	-6.7125e+01	5.3712e+02	7.7014e+01	4.6111e+01	100.0	3.3031e+01	2.2531e+02	3.6791e+02	25.0

Рисунок 4.5 — Основні статистичні дані числових характеристик у датасеті

Наступним кроком була візуалізація даних про кількість аварій у різних містах, яка зображена на рисунку 4.6. Ця інформація може допомогти дізнатися, наприклад, яким містам потрібно виділяти більші бюджети на збільшення безпеки на дорогах.

Рисунок 4.6 — Перші п’ять міст США за кількістю аварій

Також, була візуалізована інформація про кількість аварій, в залежності від погодних умов, та кількість аварій за кожен місяць, вона зображена на рисунку 4.7 і 4.8. Ця інформація допоможе зрозуміти чи є якась залежність між типом погоди та кількістю ДТП та найнебезпечніші місяці, в які треба слідкувати за дорожньою безпекою найбільше.

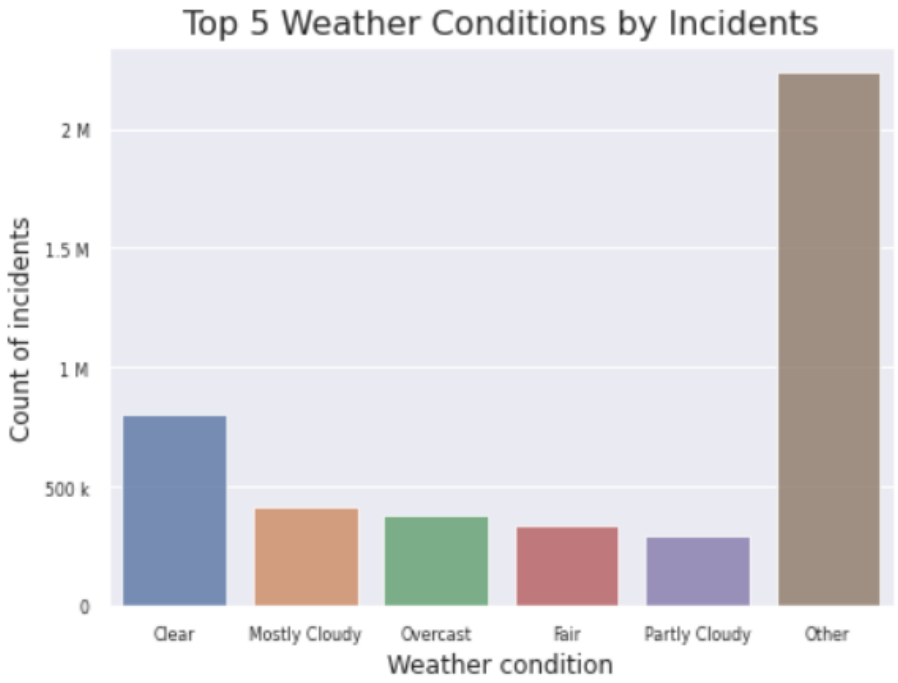


Рисунок 4.7 — Перші п’ять типів погоди за кількістю аварій

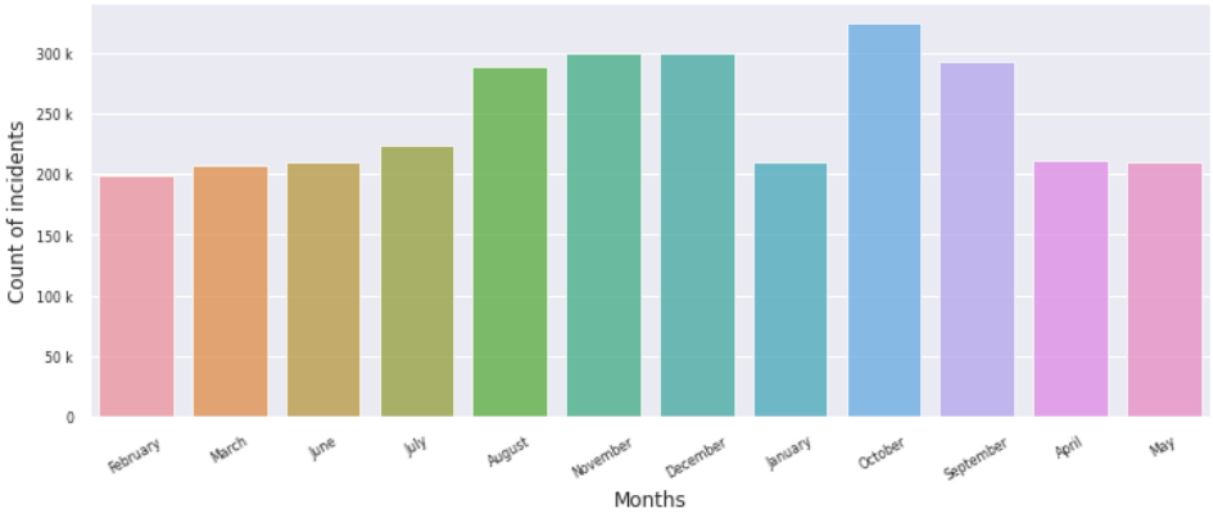


Рисунок 4.8 — Візуалізація кількості аварій за кожен місяць року

Завдяки даному етапу була отримана детальна візуалізація даних, яка дала представлення неявної, на початку, інформації про датасет та збільшила значущість результатів аналізу.

4.2.3. Прогнозування

Останній етап — етап прогнозування. Прогностична аналітика дає змогу розробити математичні моделі, щоб краще зрозуміти змінні фактори, що сприяють успіху. Прогностична аналітика спирається на формули, які порівнюють минулі успішні та невдалі приклади, а потім використовують ці формули для прогнозування майбутніх результатів [7]. Прогностична аналітика, розпізнавання шаблонів та проблеми класифікації не є новими. Вони вже довго застосовуються у фінансових сферах та страховій галузі, прогнозна аналітика полягає у використанні статистики, Data Mining та теорії ігор для аналізу поточних та історичних фактів з метою прогнозування майбутніх подій.

Значення прогнозної аналітики, наприклад, у маркетингу — очевидно. Збільшуючи розуміння поведінки та мотивації клієнтів, буде збільшуватися ефективність аналізу. Чим більше інформації про те, чому деякі клієнти лояльні, як залучати та утримувати різні сегменти клієнтів, тим більше можна розробляти відповідні переконливі повідомлення та пропозиції.

У бізнесі використовуються прогнозні моделі для оцінки ризику або потенціалу, пов’язаного з певним набором умов. Прогностичні моделі аналізують минулі показники діяльності, щоб оцінити, наскільки ймовірний клієнт проявляти конкретну поведінку в майбутньому з метою підвищення ефективності маркетингу. Фахівці з маркетингу та продажу починають фіксувати та аналізувати багато різних типів даних про клієнтів: ставлення до поведінки, поведінки та транзакцій, пов’язаних із покупними та перевагами товарів, щоб зробити прогнози щодо майбутньої поведінки щодо покупки.

Сьогодні складне середовище змушує більше організацій досліджувати прогностичну аналітику. Хоча зазвичай використовується дослідниками ринку при аналізі даних опитувань, прогнозована аналітика може бути реально застосована в таких сценаріях, як запобігання дорожньо-транспортних пригод чи планування громадського бюджету. Існує безліч способів підходу до прогнозної аналітики, і

більшість підходів залежать від чистоти бази даних, здатності видобувати дані з метою пошуку шаблонів або створення класифікацій.

На початку етапу прогнозуванню треба визначитися які стовпчики потрібні для прогнозування та як їх треба формалізувати для алгоритму. Для прогнозування кількості дорожньо-транспортних пригод потрібне поле “Start_Time”, тому що передбачення відбувається на деякий інтервал часу, та кількість інцидентів. В даній роботі кількість інцидентів можна отримати за допомогою виклику методу value_counts() до характеристики “Start_Time”. Цей метод поверне таблицю з індексом із поля “Start_Time” та полем “Count”. Далі дату треба згрупувати по дням і візуалізувати значення всієї таблиці, що зображено на рисунку 4.9.

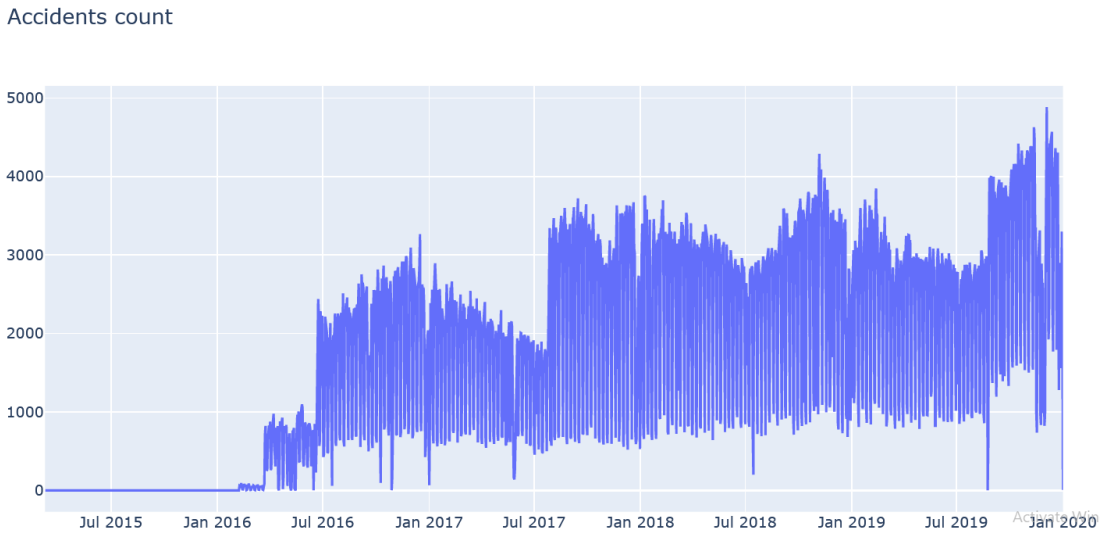


Рисунок 4.9 — Візуалізація часового ряду

Після чого, можна розрахувати кількість дорожньо-транспортних інцидентів на один день вперед, за простим принципом. Припустимо, що майбутнє значення змінної залежить від середнього n її попередніх значень, з цього випливає використання ковзної середньої, яка дорівнює 3069.5833333333335. Таке передбачення не є довгостроковим, але цей принцип можна використовувати для згладжування вхідного ряду для вияву трендів. Завдяки бібліотеці Pandas це можна зробити лише викликом Dataframe.rolling(window).mean(). Чим більше задати ширину інтервалу — тим більш згладженим виявиться тренд. При згладжуванні по днях, стає більш явною динаміка кількості інцидентів по будням і вихідним, що проілюстровано на рисунку 4.10, а при згладжуванні по тижнях — загальні зміни у кількості ДТП, що зображено на рисунку 4.11.

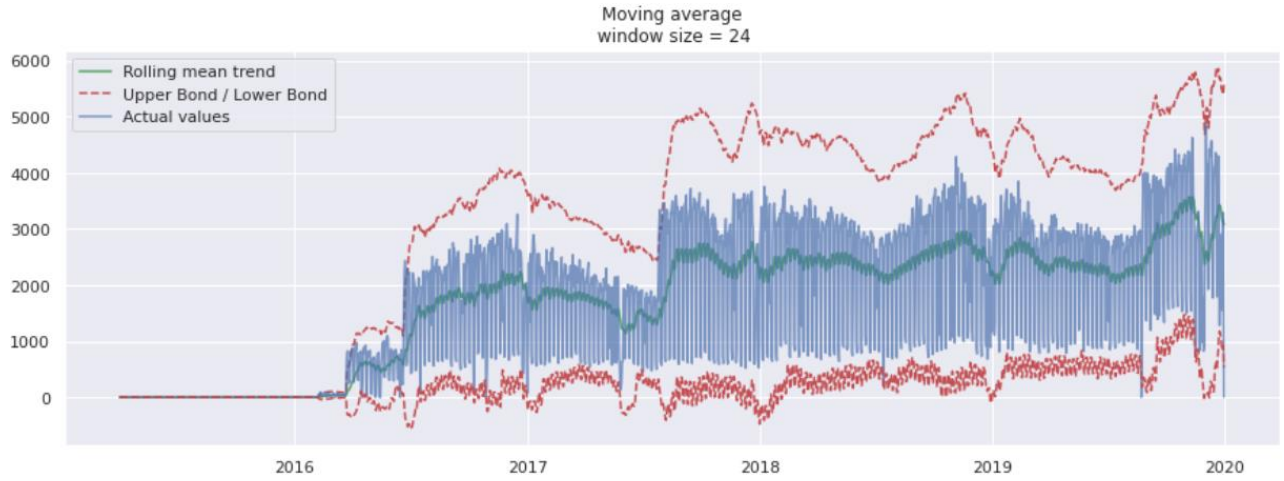


Рисунок 4.10 — Візуалізація часового ряду згладженого по дням

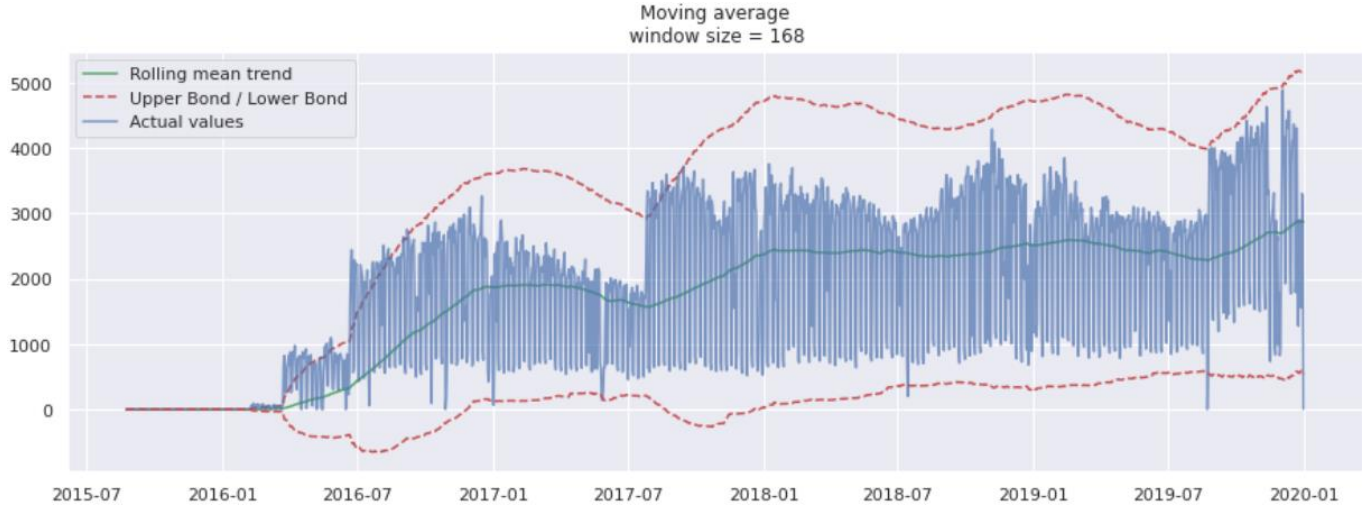


Рисунок 4.11 — Візуалізація часового ряду згладженого по тижням

Модифікацією простої ковзної середньої є зважена середня, всередині якої спостереженням надаються різні ваги, які в сумі дають одиницю, при цьому, зазвичай, останнім спостереженням присвоюється більша вага. Ковзна середня дорівнює числу 2134.96. Далі починається етап прогнозування. Для нього використовується модель ARIMA, побудовану для ряду перших різниць.

Отже, для побудови моделі треба знати її порядок. Він складається з 3-х параметрів: p — порядок компоненти AR, d — порядок інтегрованого ряду, q — порядок компоненти MA. Параметр d дорівнює числу 1, а параметри p і q треба визначити. Для того, щоб їх визначити треба проаналізувати автокореляційну (ACF) і частково автокореляційну (PACF) функції, що зображені на рисунку 4.12, для ряду перших різниць. PACF допоможе нам визначити p, оскільки по її корелограмам

можна визначити максимальний номер коефіцієнта сильно відмінний від 0 в моделі AR, а ACF допоможе визначити q, оскільки по її корелограмам можна визначити кількість автокореляційних коефіцієнтів сильно відмінних від 0 в моделі MA.

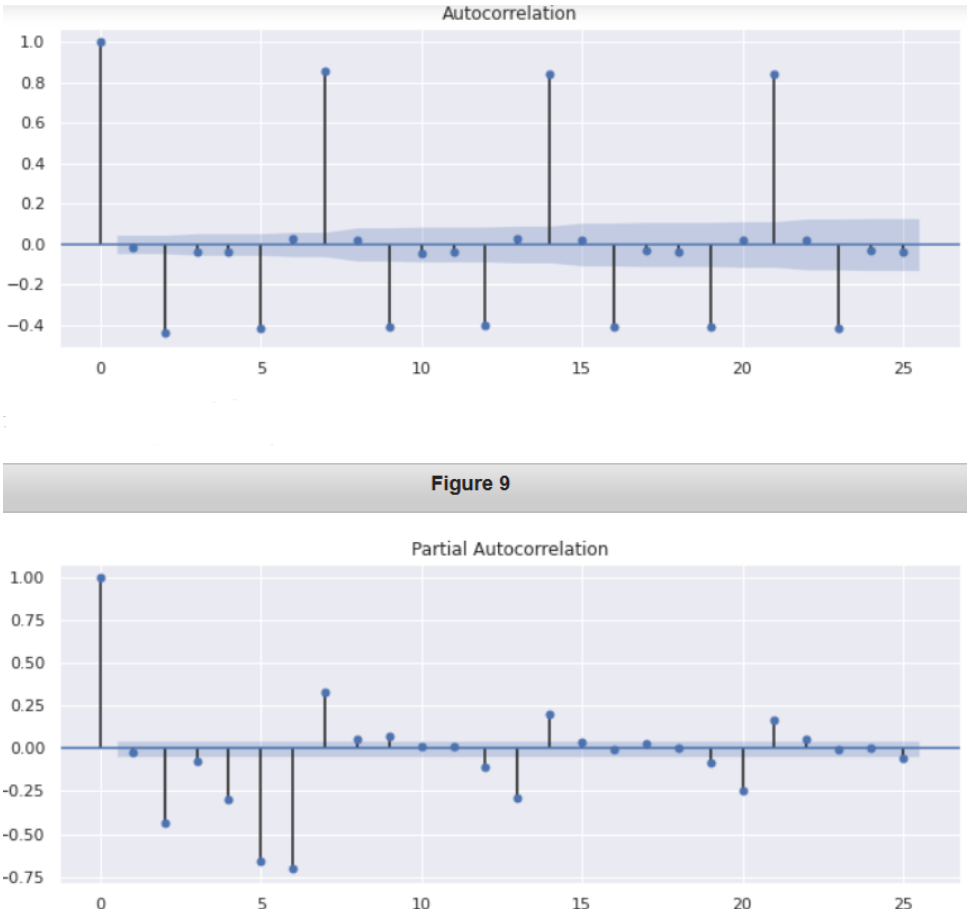


Рисунок 4.12 — Візуалізація автокореляційної та частково автокореляційної функцій

Коли відомі всі параметри можна побудувати модель, але для її побудови беруться не всі дані, а тільки частина. Дані з частини яка не потрапила в модель залишаються для перевірки точності прогнозу моделі. В результаті модель будує прогноз, який зображено на рисунку 4.13.

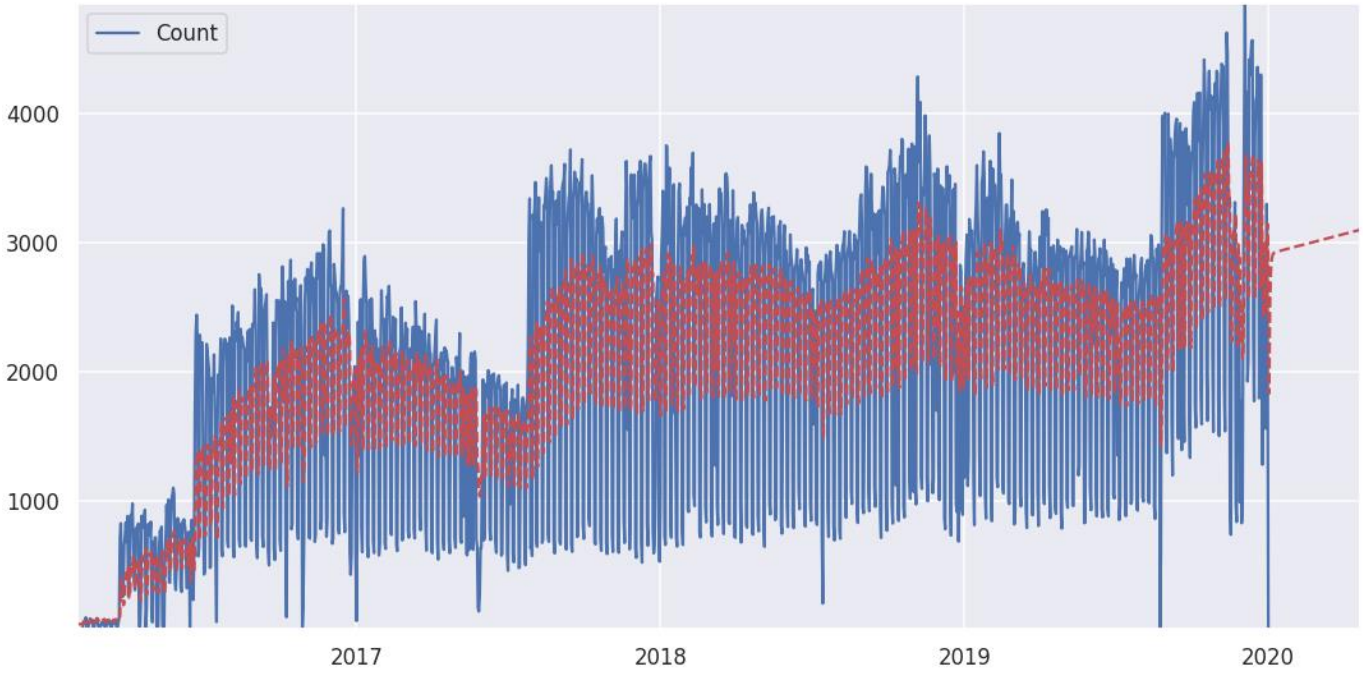


Рисунок 4.13 — Візуалізація результатів передбачення моделі ARIMA

Отже, застосовуючи ці методи до набору даних, можна отримати уявлення про сезонність та тенденції в часовому ряді. Також, можна знайти значення автокореляції в залишках моделі ARIMA, що допомагає визначити чи є модель, яка була використана — корисною.

ВИСНОВКИ

При розробці програмного продукту були отримані навички використання математичних моделей для передбачення деяких значень. Також було проведено аналіз існуючих алгоритмів передбачення, що розширило ряд можливостей та знань в проектуванні та розробці подібних сферах застосування.

Для розроблення відповідного програмного забезпечення були розв’язані наступні задачі:

1. Попередня обробка датасету.
2. Виокремлення та візуалізація значущих критеріїв датасету.
3. Проведення аналізу набору математичних моделей для здійснення найточнішого прогнозування.
4. Створення механізму прогнозування, проведення навчання математичної моделі на основі обробленого набору даних та надання отриманих результатів у зрозумілому вигляді.

Отже, як результат, було розроблено програмне забезпечення, яке дозволяє отримати проаналізовану інформацію у графічному вигляді, щодо кількості інцидентів, їх локації, а також прогнозовані місця наступних аварій з можливою кількістю дорожньо-транспортних пригод.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Di W. Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling / W. Di, A. Bhardwaj, J. Wei., 2018. – 284 c. – (Packt).

2. Freidma J. The Elements of Statistical Learning / J. Freidma, T. Robert, H. Trevor., 2009.

3. Guido S. Introduction to Machine Learning with Python: A Guide for Data Scientists / S. Guido, A. Müller., 2016. – 285 c.

4. Han, Kamber & Pei. Data Mining: Concepts and Techniques, Third Edition / Han, Kamber & Pei., 2013.

5. Heydt M. Learning Pandas – Python Data Discovery and Analysis Made Easy / Michael Heydt.

6. Kumar A. Python: Advanced Predictive Analytics / A. Kumar, J. Babcock., 2017. – 660 c. – (Packt).

7. Miller T. Modeling Techniques in Predictive Analytics with Python and R: A Guide to Data Science / Thomas Miller. – 448 c.

8. Raschka S. Python Machine Learning - Second Edition / S. Raschka, V. Mirjalili., 2017. – 622 c. – (Packt).

9. Rossant C. IPython Interactive Computing and Visualization Cookbook - Second Edition / Cyrille Rossant., 2018. – 548 c. – (Packt).

10. Wes M. Python for Data Analysis / McKinney Wes

11. Артамонов О.Ю. Моніторинг дорожнього руху / О.Ю. Артамонов, С.І. Шаповалова // Сучасні проблеми наукового забезпечення енергетики: матеріали XVIII Міжнародної науково-практичної конференції молодих вчених і студентів, Київ, 21-24 квітня 2020 р. У 2 т. - К: КПІ ім. Ігоря Сікорського, 2020.- Т.2.- С. 105.

ДОДАТОК А

Аналіз поведінки автомобіля за даними з камер спостереження

Специфікація

УКР.КПІ_ім_Ігоря_СІКОРЬКОГО_ТЕФ_АПЕПС_ТВ6121_20Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.КПІ_ім_Ігоря_СІКОРЬКОГО_ТЕФ_АПЕПС_ТВ6121_20Б_81	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.КПІ_ім_Ігоря_СІКОРЬКОГО_ТЕФ_АПЕПС_ТВ6121_20Б _12-1	Assident-Analyze.ipnb	Основний компонент
УКР.КПІ_ім_Ігоря_СІКОРЬКОГО_ТЕФ_АПЕПС_ТВ6121_20Б_12-1	main.py	Основний компонент
УКР.КПІ_ім_Ігоря_СІКОРЬКОГО_ТЕФ_АПЕПС_ТВ6121_20Б_12-1	accident.py	Основний компонент

ДОДАТОК Б

Аналіз поведінки автомобіля за даними з камер спостереження

Текст програми

УКР.КПІ_ім_Ігоря_СІКОРЬКОГО_ТЕФ_АПЕПС_ТВ6121_20Б

Аркушів 8

Київ 2020

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
%matplotlib inline

import folium
from folium import plugins
import seaborn as sns

import calendar
import category_encoders as ce

from sklearn import ensemble
import xgboost as xgb

# Set general settings for Pandas
pd.set_option('display.max_columns', 100)
pd.set_option('display.max_rows', 100)
pd.set_option('display.precision', 4)

# Read the whole dataset into a Pandas' DataFrame
accidents = pd.read_csv('datasets/dataset_3/us_accidents_dec_19.csv', encoding='utf-8',
low_memory=False, parse_dates=['Start_Time'])

# Get briefly info about dataset
accidents.info(null_counts=True)

# Get percentage of missing values in each column
fig = plt.figure(figsize=(20,10))

# Calculate percentage
missing_percentage = (100 * accidents.isnull().sum() / accidents.shape[0]).round(2)
```

```
# Get columns which have a percentage of missing values greater than 0
missing_percentage = missing_percentage[missing_percentage.values > 0].sort_values(ascending=False)

# Show plot
plot = sns.barplot(x=missing_percentage.index, y=missing_percentage.values)
plt.xticks(rotation='vertical')

# Delete top 5 columns with biggest percentage of missing values
del accidents['End_Lat']
del accidents['End_Lng']
del accidents['Precipitation(in)']
del accidents['Number']
del accidents['Wind_Chill(F)']

# Delete columns with unusable info
del accidents['Airport_Code']
del accidents['Street']
del accidents['Zipcode']
del accidents['ID']
del accidents['Source']
del accidents['Civil_Twilight']
del accidents['Nautical_Twilight']
del accidents['Astronomical_Twilight']
del accidents['Country']
del accidents['End_Time']
del accidents['Weather_Timestamp']

newColumns = {
    'Temperature(F)': 'Temperature_F',
    'Humidity(%)': 'Humidity_Pct',
    'Pressure(in)': 'Pressure_In',
    'Visibility(mi)': 'Visibility_Mi',
    'Wind_Speed(mph)': 'Wind_Speed_Mph',
    'Distance(mi)': 'Distance_Mi'
}
```

```
accidents.rename(columns=newColumns, inplace=True)

accidents.Start_Lat = accidents.Start_Lat.astype('float16')
accidents.Start_Lng = accidents.Start_Lng.astype('float16')
accidents.Distance_Mi = accidents.Distance_Mi.astype('float16')
accidents.Temperature_F = accidents.Temperature_F.astype('float16')
accidents.Humidity_Pct = accidents.Humidity_Pct.astype('float16')
accidents.Pressure_In = accidents.Pressure_In.astype('float16')
accidents.Visibility_Mi = accidents.Visibility_Mi.astype('float16')
accidents.Wind_Speed_Mph = accidents.Wind_Speed_Mph.astype('float16')
accidents.Description = accidents.Description.astype('string')
accidents.Timezone = accidents.Timezone.astype('string')
accidents.City = accidents.City.astype('string')
accidents.Side = accidents.Side.astype('string')
accidents.Wind_Direction = accidents.Wind_Direction.astype('string')
accidents.Sunrise_Sunset = accidents.Sunrise_Sunset.astype('string')
accidents.County = accidents.County.astype('category')
accidents.State = accidents.State.astype('category')
accidents.Weather_Condition = accidents.Weather_Condition.astype('category')
accidents.Severity = accidents.Severity.astype('int16')

newColumns = {
    'Temperature_F' : 'Temperature_C',
    'Visibility_Mi': 'Visibility_Km',
    'Wind_Speed_Mph': 'Wind_Speed_Mps',
    'Distance_Mi': 'Distance_Km'
}

def convert_f_to_c(f):
    return (f - 32) * 5.0/9.0

def convert_mi_to_km(mi):
    return mi * 1.609344
```

```
accidents.Wind_Speed_Mph = accidents.Wind_Speed_Mph.apply(lambda mph: mph *
0.44704).astype('float16')
accidents.Temperature_F = accidents.Temperature_F.apply(convert_f_to_c).astype('float16')
accidents.Visibility_Mi = accidents.Visibility_Mi.apply(convert_mi_to_km).astype('float16')
accidents.Distance_Mi = accidents.Distance_Mi.apply(convert_mi_to_km).astype('float16')

accidents.rename(columns=newColumns, inplace=True)

fig, axes = plt.subplots(1, 2, figsize=(15,5))

cities = accidents.City.astype('string').value_counts()
top_cities = cities.head()
top_cities['Other'] = cities[:5].sum()

ax1 = sns.barplot(top_cities.index, top_cities.values, alpha=0.8, ax=axes[0])

ax1.yaxis.set_major_formatter(ticker.EngFormatter())
ax1.set_title('Top 5 US Cities by Incidents', fontsize=16)
ax1.set_xlabel('City', fontsize=12)
ax1.set_ylabel('Count of incidents', fontsize=12)

weather = accidents.Weather_Condition.astype('string').value_counts()
top_weather = weather.head()
top_weather['Other'] = weather[:5].sum()

ax2 = sns.barplot(top_weather.index, top_weather.values, alpha=0.8, ax=axes[1])

ax2.yaxis.set_major_formatter(ticker.EngFormatter())
ax2.set_title('Top 5 Weather Conditions by Incidents', fontsize=16)
ax2.set_xlabel('Weather condition', fontsize=12)
ax2.set_ylabel('Count of incidents', fontsize=12)

plt.show()

fig = plt.figure(figsize=(12, 10))
```

```
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(212)

years = accidents.Start_Time.map(lambda date_time: date_time.year)
plot1 = sns.countplot(years, alpha=0.8, ax=ax1)

plot1.set_xlabel('Year', fontsize=12)
plot1.set_ylabel('Count of incidents', fontsize=12)
plot1.yaxis.set_major_formatter(ticker.EngFormatter())

months = accidents.Start_Time.map(lambda date_time: calendar.month_name[date_time.month])
plot2 = sns.countplot(months, alpha=0.8, ax=ax2)

plot2.set_xlabel('Months', fontsize=12)
plot2.set_xticklabels(plot2.get_xticklabels(), rotation=30)
plot2.set_ylabel('Count of incidents', fontsize=12)
plot2.yaxis.set_major_formatter(ticker.EngFormatter())

plt.show()

states = accidents.State.unique()

def get_count_of_accidents(severity, state):
    condition = (accidents.Severity == severity) & (accidents.State == state)
    return accidents[condition].count()['Id']

severity_1_by_state = [get_count_of_accidents(1, state) for state in states]
severity_2_by_state = [get_count_of_accidents(2, state) for state in states]
severity_3_by_state = [get_count_of_accidents(3, state) for state in states]
severity_4_by_state = [get_count_of_accidents(4, state) for state in states]

plt.figure(figsize=(18,12))

plt.xlabel('States')
```

```
plt.ylabel('Count')

plt.bar(states, severity_1_by_state, label='Severity 1')
plt.bar(states, severity_2_by_state, label='Severity 2')
plt.bar(states, severity_3_by_state, label='Severity 3')
plt.bar(states, severity_4_by_state, label='Severity 4')

plt.legend()

TMC_counts = accidents.TMC.value_counts()
plt.figure(figsize=(16, 10))

ax = sns.barplot(TMC_counts.index, TMC_counts.values, order=TMC_counts.index)
ax.set(xlabel='TMC type', ylabel='Amount')
ax.yaxis.set_major_formatter(ticker.EngFormatter())
TMC_counts = accidents.TMC.value_counts()
plt.figure(figsize=(16, 10))

ax = sns.barplot(TMC_counts.index, TMC_counts.values, order=TMC_counts.index)
ax.set(xlabel='TMC type', ylabel='Amount')
ax.yaxis.set_major_formatter(ticker.EngFormatter())

tmc_table = pd.read_html('https://wiki.openstreetmap.org/wiki/TMC/Event_Code_List', attrs={"class":
"wikitable"})
tmc_table = tmc_table[0]

t = pd.DataFrame()
t['Code'] = TMC_counts.index
t['Count'] = TMC_counts.values

t1 = pd.DataFrame({'Code': tmc_table.Code, 'Description': tmc_table.Description})

t.set_index('Code').join(t1.set_index('Code'))
lats = list(accidents.Start_Lat[::100])
longs = list(accidents.Start_Lng[::100])
```

```
places = [[x[0], x[1]] for x in zip(lats, longs)]

m = folium.Map(places[0], tiles='OpenStreetMap', zoom_start=13)
plugins.MarkerCluster(places).add_to(m)

import statsmodels.api as sm
from statsmodels.iolib.table import SimpleTable
from sklearn.metrics import r2_score
import ml_metrics as metrics
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

data_frame = accidents.Start_Time.value_counts().sort_index()
time_series = pd.DataFrame(data_frame.resample('D').std().dropna())
time_series.rename(columns={'Start_Time': 'Count'}, inplace=True)
sns.set(rc={'figure.figsize':(11, 4)})
sns.lineplot(data=time_series, x=time_series.index, y=time_series.Count)
```

ДОДАТОК В

Аналіз поведінки автомобіля за даними з камер спостереження

Опис програми

УКР.КПІ_ім_Ігоря_СІКОРЬКОГО_ТЕФ_АПЕПС_ТВ6121_20Б

Аркушів 9

Київ 2020

АНОТАЦІЯ

Дана програмна система містить опис системи що розроблена для отримання даних аналізу та передбачень основаних на наборі даних про дорожньо-транспортні пригоди. Вона складається з двох модулів. Перший модуль відповідає за запуск сервера, а інший – містить у собі опис аналізу набору даних та деяких передбачень на його основі. Ця система створена в Jupyter з використанням мови програмування Python.

Функціональність веб-системи реалізована за допомогою фреймворку Flask для розробки веб-додатків.

В додатку виконуються такі функції

— автоматичне завантаження вхідних даних

— аналіз вхідних даних

— візуалізація результатів аналізу

— передбачення на основі вхідних даних

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У даному додатку описано систему аналізу набору даних та розрахунку передбачення на його основі. У додатку Б міститься програмний код компоненту.

Розроблена програмна система працює у вигляді веб-додатку, тому доступна на будь-якій операційній системі. Для локальної розгортки системи, необхідно встановити Python 3.6, фреймворк Flask та Docker.

Додаток розроблений за допомогою мови програмування Python та фреймворку Flask.

ФУНКЦІНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний додаток надає візуалізовані статистичні дані значущих критеріїв датасету та результати передбачень розраховані за допомогою методів і алгоритмів Data Mining.

Розроблена система рекомендована для використання менеджерами приватних фірм та державних структур які займаються регулюванням безпеки на дорогах для отримання більш об’ємної інформації щодо управління дорожньо-транспортною ситуацією.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Дана система розроблена за архітектурою “Клієнт-Сервер”. На початку роботи програми, користувач відправляє HTTP запит, та у відповідь на нього отримує всі дані які були отримані внаслідок аналізу та розрахунків на основі вхідного датасету. Відповідь генерується з шаблону із залученням HTML, CSS та JavaScript, який реалізований на мові “Jinja 2”. Також, на сервері реалізоване API для отримання результатів роботи програми.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для розробки даної програмної системи було обрано Jupyter (мова імплементації — Python).

Розроблений додаток — кросплатформний, тому він працює на всіх відомих операційних системах. Також, він не потребує встановлення додаткових компонентів.

Для використання розробленого додатку користувач повинен мати персональний комп’ютер або мобільний пристрій із встановленим браузером, що підтримує JavaScript.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програмний додаток не потребує інсталяції. Для роботи з програмою достатньо відкрити браузер на сторінці додатку.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідні дані для додатку:

1) Датасет про дорожньо-транспортні інциденти у форматі *.csv

Вихідні дані додатку:

1) Візуалізовані статистичні дані значущих критеріїв датасету;
2) Результати передбачень на основі даного датасету.